PROGRAMMATION ORIENTÉE OBJET - INTERFACES GRAPHIQUES

RAPPORT DE PROJET

Groupe 10
BETRAOUI Mohammed Ali
HADDAG Amayas

1 Introduction

Le projet de POO-IG est un projet de programmation en Java dont le but est d'évaluer les connaissances en matière de notions de programmation orientée objet. Le thème couvert par ce projet est les jeux-vidéos de type "Tower-Defense", un type de jeu dont le but est de défendre sa base contre des ennemis en construisant des tours qui les attaqueront. Le jeu se déroule généralement sur un plateau fixe où toutes les interactions se font.

Le but étant de proposer une implémentation complète pour un jeu de type "Tower-Defense", nous étions munis d'un cahier de charges (minimal et avancé) couvrant les ensembles des tâches à réaliser, restrictions à respecter et ajouts potentiels à l'implémentation du jeu. Il est à préciser qu'il nous est aussi laissé carte blanche quant à ce qu'il s'agit des touches personnelles et explorations de nouvelles idées.

Ce rapport représente un résumé de l'ensemble des tâches du cahier des charges traîtées ainsi que du processus d'implémentation de notre version du jeu.

2 Choix du type du jeu

Dans le cahier des charges fourni, nous avons eu le droit à deux potentielles suggestions d'implémentation différentes pour le jeu, chacune avec ses propres propriétés :

- La première implémentation est inspirée du célèbre jeu **Plants vs. Zombies** où les ennemis avancent sur plusieurs lignes droites et les tours sont posées directement sur le chemin.
- La seconde implémentation est tirée du jeu **Bloons TD** où la carte du jeu représente un chemin sur lequel les ennemis avancent et les tours sont posées en dehors de ce chemin.

Nous avons décidé de choisir la deuxième implémentation pour notre version du jeu.

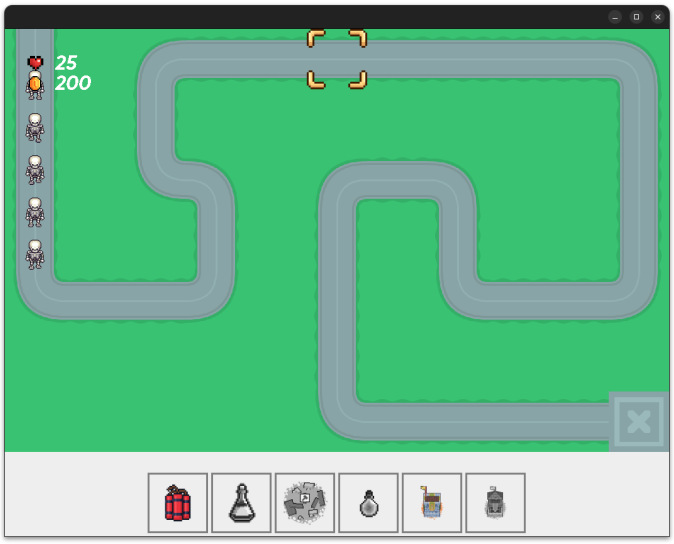
Ainsi, le jeu consiste en une carte ayant un chemin tracé jusqu'à la base du joueur, le chemin peut potentiellement avoir des intersections, les ennemis marchent le long de ce chemin. Le joueur quant à lui, pose des tours et des items afin d'empêcher ces ennemis d'arriver à sa base.

Le joueur est alors perdant si sa base ne possède plus aucun point de vie, ces points de vie seront retirés par les ennemis qui auraient réussi à atteindre la base.

3 Présentation du jeu

Le jeu se divise en trois parties majeures :

3.1 La carte



La carte est la section du jeu qui représente le plateau sur lequel toutes les interactions se font, elle est composée principalement d'un chemin possédant un point de départ et un point d'arrivée. Les ennemis apparaissent sur le point de départ, parcourent le chemin afin de tenter d'arriver au point d'arrivée qui représente la base du joueur. S'ils réussissent, ils infligeront des dégâts à ce dernier et pourra potentiellement perdre.

Le reste de la carte représente l'espace sur lequel le joueur pourra poser ses tours qui attaqueront les ennemis.

Le joueur possède à sa disposition également des items qui seront posés directement sur le chemin des ennemis comme des pièges à dégâts, des bombes, du poison ...

3.2 L'inventaire

L'inventaire est un ensemble de cases (boutons) qui représentenent :

- Des tours :
    1. La tour `SimpleTower` : attaque les ennemis un par un, en leur tirant dessus sur un rayon de deux cases avec une cadence moyenne.
    2. La tour `BombTower` : attaque les ennemis en lançant une bombe explosive sur eux dans un rayon d'une case autour d'elle, elle possède également une cadence assez lente.
- Des Items :
    1. La `Bomb` : peut être posée sur le chemin des ennemis, une fois posée, elle déclenche une explosion de rayon de 1 case autour d'elle qui inflige des dégâts aux ennemis.

2. Le `Freeze` : permet de geler les ennemis sur lesquels elle est lancée et les empêche de bouger durant une période de temps de 5 secondes.
3. Le `Poison` : une fois lancé, il contamine les ennemis et leur inflige des dégâts minimes en continu durant 3 secondes.
4. Le `Trap` : représente un piège à dégâts qui peut être posé sur le chemin des ennemis, chaque ennemi qui marche dessus subit des dégâts et meurt instantanément.

3.3 Le Shop

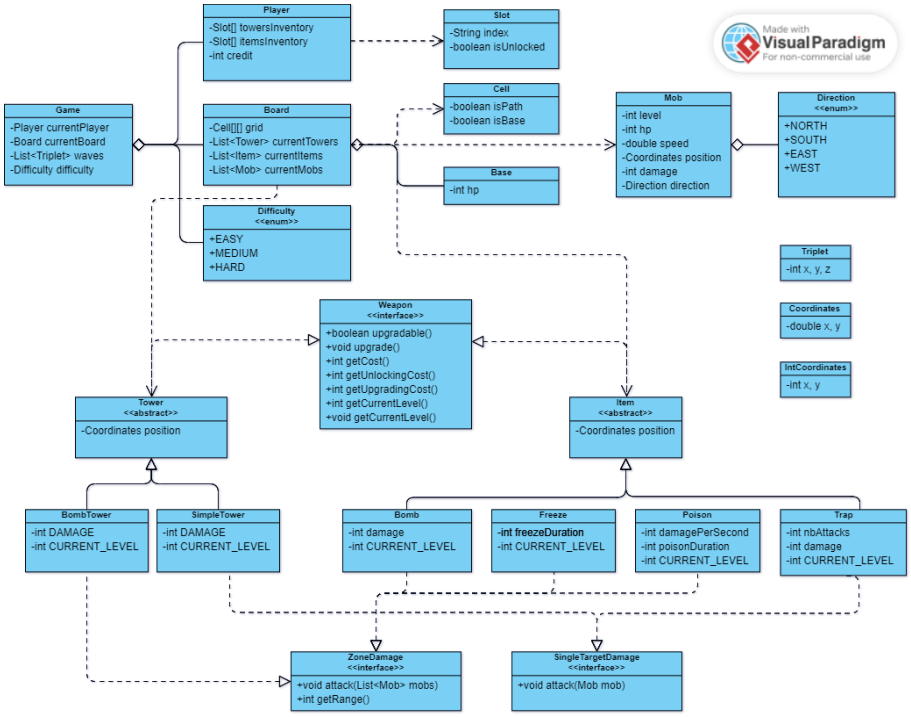
Le shop est une section du jeu dans laquelle l'utilisateur peut :

- Débloquer de nouvelles tours ainsi que de nouveaux items, qu'il achète avec ses crédits, grâce au bouton "UNLOCK".
- Améliorer les éléments de son inventaire les rendants plus puissants et plus efficaces grâce au bouton "UPGRADE".

4 Détails d'implémentation

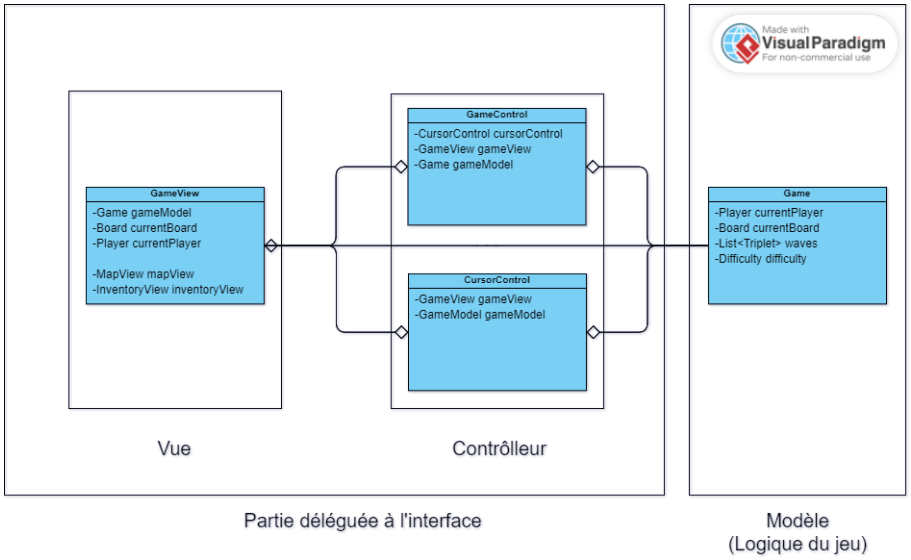
4.1 Modèle

Le diagramme UML ci-dessous représente un schéma détaillé de l'implémentation de la logique du jeu, les relations des classes `Coordinates`, `IntCoordinates` et `Triplet` ne sont pas représentées étant donné qu'elles s'utilisent dans pratiquement toutes les classes.



4.2 Interactions Model-View-Control

Le diagramme UML ci-dessous représente l'architecture Model-View-Control utilisée dans notre version du jeu ainsi que les relations entre les classes des trois catégories.

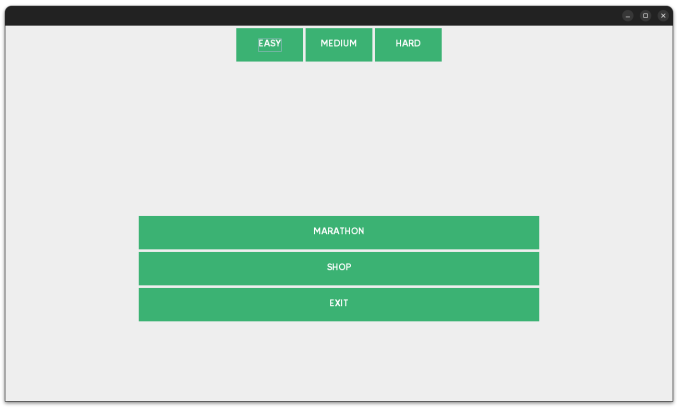


5 Respect du cahier des charges

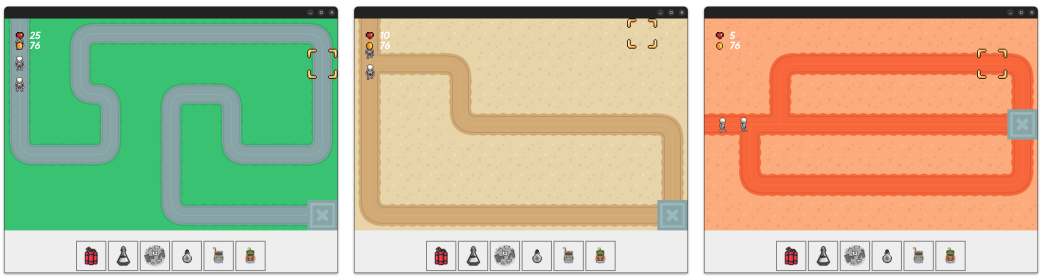
5.1 Charges minimales

5.1.1 Environnement de jeu

Dès qu'on lance le jeu on se retrouve dans le menu du jeu qui nous permet de choisir :



- Niveaux de difficulté :
  - Facile : Nous avons 4 vagues d'ennemis, chaque vague ayant un nombre réduit de mobs de bas niveaux. Il n'y a qu'un seul chemin assez long pour les mobs.La base est assez résistante.
  - Moyen : Cette fois nous avons 7 vagues d'ennemis. Les vagues sont plus imposantes : plus d'ennemi avec des niveaux plus haut. Les mobs ont deux chemins vers la base rendant la tâche plus difficile au tours de défendre une base avec moins de points de vies.
  - Difficile : Trois chemins très court vers une base avec très peu de points de vie. Dix vagues d'ennemis des plus redoutables.
- Chemins et décors : Nous avons choisit un décor différent pour chaque niveau de jeu
  - Un décors avec de la verdure pour le niveau facile.
  - Un décors avec comme thème le désert pour le niveau moyen.
  - Un décors dans une terre en feu pour le niveau difficile.



- Deux modes de jeu :
  - Mode de jeu normal : avec trois niveaux différents comme cité précédemment.La partie se finit par une victoire si on resiste à toutes les vagues. Ou bien par une défaite si notre base tombe.
  - Mode Marathon : un mode de jeu avec des vagues à l'infini. Le nombre de mobs ainsi que leur niveau croît de façon exponentielle.

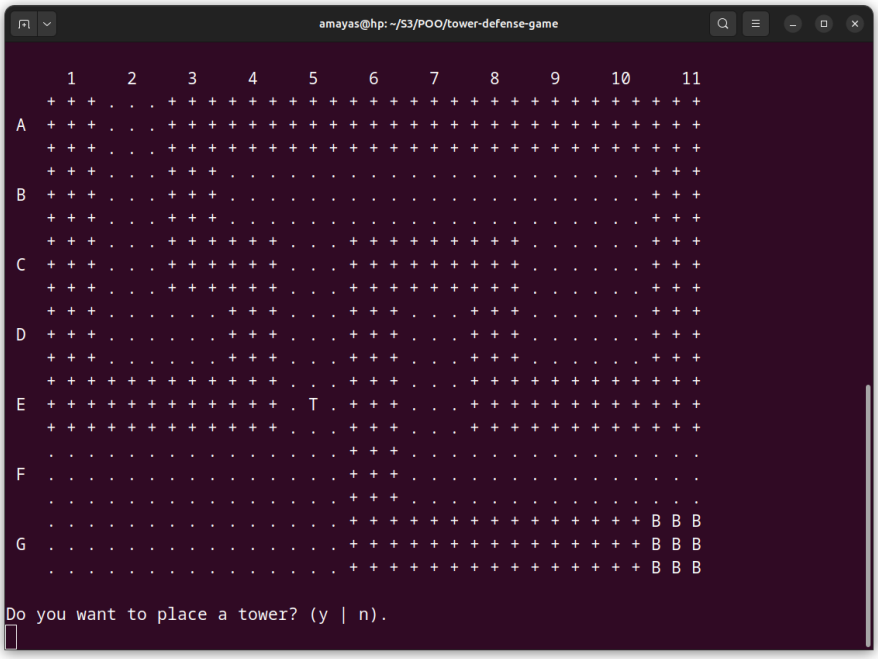
5.1.2 Implémentation des règles générales

1. Un point de départ d'où les ennemis apparaissent qui change en fonction de la partie.

2. Un point d'arrivée : les cordonnées de notre base qui eux aussi changent en fonction de la partie.
3. Plusieurs types d'ennemis : Nous en avons trois. Avec des vitesses, barres de vies et puissances différentes.
4. Plusieurs types de tours : Des tours simple avec des attaques rapides, à longue porté. Ainsi que des tours de bombe qui eux sont lentes, à courte portée.
5. Diversité des tours : Nous retrouvons des tours de type `SingleTargetDamage` qui attaque un ennemi à la fois. Ainsi que des tours de type `ZoneDamage` qui eux infligent des dégâts de zone. Nous avons également des `Item` avec plusieurs fonctions différentes : ralentir l'ennemi, l'empoisonner, ...etc.
6. Ressources : lorsqu'on tue un ennemi, on recupère des crédits qui servent à placer des tours ou des items, à en débloquer des nouveaux et mêmes à améliorer ces derniers.
7. Progression : Chaque vague a plus de mobs que celle qui la précéde avec des mobs de niveaux plus hauts.

5.1.3 Une version textuelle du jeu

Le jeu possède une version textuelle pouvant se jouer entièrement dans le terminal indépendamment de l'interface graphique, le mode console est une version très simplifiée du jeu, qui reprend le mode facile de la version graphique, dans ce mode, le joueur peut poser les tours de type `SimpleTower` qui attaqueront les ennemis qui arriveront sur le chemin. La figure ci-dessous représente une visualisation de ce mode :



Dans ce mode, les tours sont représentées par la lettre `T` et les ennemis par des numéros qui représentent leurs points de vie.

Les instructions de la partie sont dictées directement par un message dans le terminal auquel le joueur devra répondre à chaque fois.

5.1.4 Une version graphique du jeu

Un mode de jeu graphique fait en utilsant les bibliothèques `swing` et `awt`.
On peut jouer au jeu en utilisant la souris. On y retrouve même des animations.

5.2 Fonctionnalités avancées du jeu

5.2.1 Sauvegardes

Grâce à l'interface `Serializable`, le jeu peut sauvegarder la progression du joueur. Lorsqu'un joueur ferme le jeu puis le relance : son nombre de crédit, les objets débloqués ainsi que le niveau de ses tours et de ses items ne sont pas perdus.

Les données du joueur sont stockées dans des fichiers `.txt` disponibles dans le répertoire `data` du code source du jeu.

5.2.2 Améliorations

Comme indiqué précédemment, le Shop du jeu possède la fonctionnalité d'amélioration de contenu, le joueur est capable en ayant le credit requis d'améliorer ses tours et ses items, cela leur confie plus de puissance et d'efficacité.

5.2.3 Débloquage de contenu

Il est également possible de débloquer de nouvelles tours, de nouveaux items en échange de crédit.

6 Difficultés rencontrées

Une des difficultés rencontrées durant la création de notre jeu était lors de l'implémentation du système de sauvegarde des données des joueurs.

Cette fonctionnalité utilisait des méthodes des classes `ObjectInputStream` et `ObjectOutputStream`. Notamment la méthode `readObject` qui renvoie les données sérialisées dans un fichier sous le type `Object`.

Afin de stocker les niveaux d'amélioration associés à chaque tour et item, nous avons utilisé une table associative de type `Map<String, Integer>` qui associe à tout index de tour ou item (de type `String`) son niveau actuel.

Etant donné que la lecture de ces données depuis leur fichier correspondant renvoie un objet de type `Object`, il était nécessaire de le downcast en type `Map<String, Integer>`. Puisque la classe `Map` est générique, le casting engendrait un warning de type `unchecked` lors du cast, ceci même en faisant en sorte de vérifier la compatibilité du type générique manuellement.

Après quelques séances de recherche concernant la résolution de ce problème, nous sommes arrivés à la conclusion que la majorité des solutions proposées depuis internet ignorent le warning avec l'instruction `@SupressWarning("unchecked")`.

7 Pistes d'extension non-implémentées

7.1 Base de données des joueurs

Jusqu'ici le jeu n'est capable de stocker qu'un seul profil de joueur par machine, ce processus est fait automatiquement. Une façon d'améliorer cette fonctionnalité aurait été de modéliser une base de données des joueurs avec un système d'authenficiation simplifié qui permettrait au joueur de se connecter avec plusieurs profils et par conséquent, avoir plusieurs progressions différentes en parallèle.

8 Conclusion

Le projet de POO-IG constitue une application intéressante des concepts vus durant le cours de Programmation Orientée Objet. Il permet également de mieux comprendre les design-patterns les plus récurrents, notamment l'architecture Model-View-Control.

Le thème du projet était également un choix perninent étant donné qu'il permet aux étudiants de couvrir la majorité des concepts fondamentaux de la POO. Il leur permet également l'exploration de nouvelles pistes, de nouveaux concepts et de nouvelles idées d'implémentation comme nous l'avions vu précédemment avec la sérialisation de données.