**Tower Defense**

**Nous avons programmer le jeu en java avec swing sur l’éditeur de texte Visual Studio Code. Pour lancer le jeu cliquer sur le fichier exe.**

**Modèle Vue Controller :**

Le projet est organisé en MVC. Les modèles permettent de définir les attribut d’un objet ainsi que les fonctions qui seront appelés. La vue dessine le terrain de jeu ainsi que les images et les tires. Le controller permet d’interagir et d’envoyer les informations entre les modèles et la vue. Il sert d’intermédiaire entre la vue et le modèle.

Modèles :

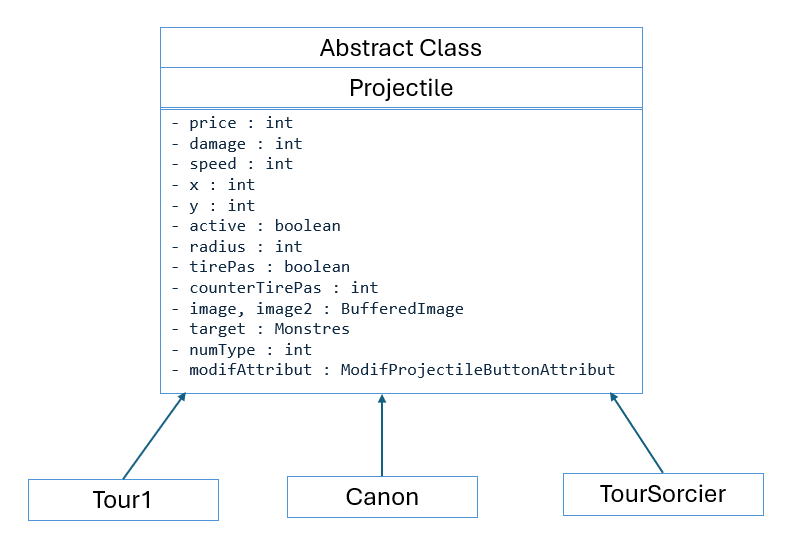
Projectile (Figure 1) : classe abstraite qu’héritent les classes Tour1, Canon et TourSorcier. Cette classe regroupe tous les attributs communs. 

Figure 1 Diagramme Projectile et ses classes filles

Musique : lancer ou stopper la musique de fond.

ShotSounds : bruits pour les tires, tous differents.

Dans notre programme nous avons comme Controller :

MouseController : interactions utilisateur/souris. Regroupe toutes les actions déclenché lorsque l’utilisateur clique, presse, glisse et relache la souris. Lorsqu’il veux sélectionner une tour à acheter s’il a assez d’argent pour ensuite la déposer à un endroit accessible, il doit sélectionner un projectile dans la barre d’inventaire pour ensuite la faire glisser jusqu’à un endroit de la carte. L’utilisateur peux aussi cliquer sur un projectile disposé dans la map pour faire une amélioration ou vendre le projectile.

KeyboardController : interactions utilisateur/clavier :

P ou p : pause

Q ou q : quitter

m : désactiver la musique

M : activer la musique

TourController :

Stocke tous les projectiles placés sur la carte du jeu et vérifie leurs placements corrects pour qu’ils ne soient pas en dehors de la fenêtre de jeu.

PathController :

Vérifie si un projectile est sur le chemin du niveau avec les coordonnées.

BarreInventaire :

Dispose automatiquement tout les types de projectiles dans la barre d’inventaire en calculant leur positions par rapport à la largeur de l’écran, l’espacement et au nombre de projectile à afficher. Cela va permettre de choisir parmis la tour, le canon ou la tour de sorcier.

**Fonctionnalités du jeu :**

* Affichage du rayon en bleu si le projectile peux être placé à un endroit, en rouge sinon
* Amélioration des projectiles si assez d’argent : 3 niveaux d’amélioration
  + Tour : rayon et dégats
  + Canon : dégats et vitesse de tire
  + Tour Sorcier : rayon
* Vendre projectile et l’enlever de la carte
* Mettre en pause le jeu avec la touche p ou P
* Quitter la partie une fois cliqué sur start dans la page d’accueil avec la touche q
* Musique : possibilité de la stopper avec la touche m du clavier et de la relancer avec la touche M

**Design Pattern :**

Iterator :

Génération de monstres

Factory : pour avoir une modularité et une flexibilité dans la conception des tours du jeu, nous avons utilisé la Factory qui a permis de regrouper touts les types de projectiles.

L’énumération ProjectileType.java regroupe les types : TOUR, CANON et TOUR\_SORCIER

La Factory ProjectileFactory créer une instance de projectile selon le type énuméré en paramètre d’une fonction.

Cela est utilisé dans le controller MouseController lorsque l’utilisateur intéragit avec la souris pour sélectionner un des projectiles à créée. Par exemple avec le choix de la tour dans la barre d’inventaire la Factory utilise le type TOUR pour créer une instance de Tour1.java avec ses paramètres lorsqu’elle est déposé sur le plateau de jeu.

Strategy :

L’interface Strategy est commune à toutes les stratégies de tire pour chaque projectile. Elle déclare une méthode que les projectiles utilisent pour leur tire.

DirectShoot, RallentirShoot et ZonedShoot impémentent la fonction de l’interface mais de différentes manières.

Chaque projectile créer un objet de ShootStrategy et applique le type de tire sur les monstres.

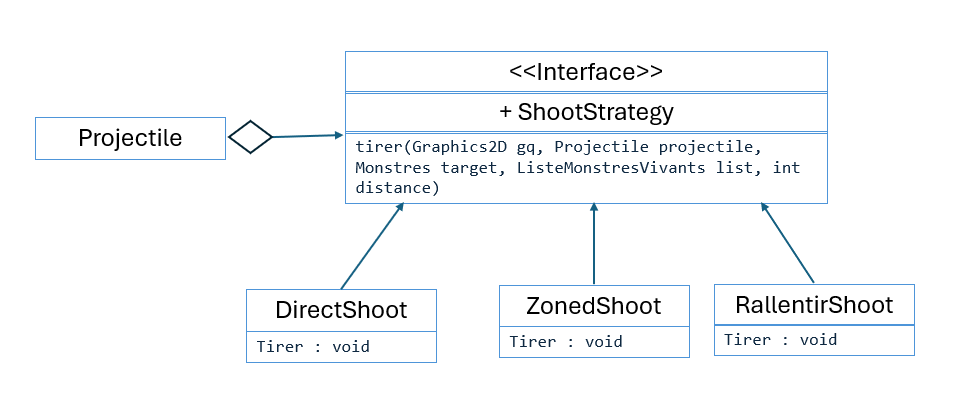
 Cela permet de regrouper tous les tires possibles en les mettant dans des classes séparées et de rendre leurs objets interchangeables : tour fait des dégats seulement sur un personnage a la fois . Canon fait des dégats de zone. Tour sorcier ralentis les personnage dans un certain rayon.

Figure 2 Design Pattern Strategy

Singleton : chateaux/pyramide a la fin du chemin est représenté comme singleton car il est unique dans le jeu.

Tunel.java dans notre jeu créer une instance unique pour la tour avec ses vies qui diminuent lorsqu’un monstre entre dans cette zone.

Lorseque le jeu est au niveau 2, le nombre de vies du tunel est égale au nombre de vie de la fin duu niveau 1.

Abstract class