

# Projet de Jeux S2 :

## La revanche des Loups

Par :  
*Liu Christophe*  
*Yahia Mohcine*  
*El Mestari Bilel*

Sommaire
----------

1) <i>Document utilisateur</i> .....	3/9
<i>Contexte</i> .....	3/9
<i>Objectif</i> .....	3/9
<i>Modèle</i> .....	4/9
2) <i>CPOO</i> .....	5/9
<i>Architecture</i> .....	5/9
<i>Diagramme de Classe</i> .....	6/9
<i>Diagramme de Séquence</i> .....	7/9
<i>JUnit et Exception</i> .....	8/9
<i>Structure de données</i> .....	8/9
<i>Utilisation maîtrisé Algorithmes intéressants</i> .....	9/9

## **Document utilisateur** *(Liu Christophe)*

C'est un Tower Défense, à laquelle le joueur doit défendre un objectif (la maison dans notre cas) contre des vagues d'ennemis (les loups). Pour cela le joueur dispose de 3 différents types de tour, avec sa propre valeur, pour que le joueur dépense son argent pour se défendre, en gagnant à chaque fin de vague.

### **Contexte :**

Le jeu reprend le thème "Les trois petit cochons", une armée de grands méchants Loups attaque la maison des cochons pour prendre leurs revanches et pour les manger. Les cochons vont se défendre avec les tours qui sont en paille, bois et brique. Le but des cochons sera d'organiser leurs forteresses de tel sorte que les loups ne puissent pas les dévorer.

Les loups viennent de la droite et se dirigent vers la maison principale des cochons (qui a un certain nombre de pv pour l'attaquer), qui se situe à l'extrémité gauche, pour le détruire. Les loups, avant de faire face aux tours, devront contourner les obstacles de la map (rocher, rivière...). Les tours tirent sur un loup dès qu'il rentre dans son périmètre de tir et ne le lâche que lorsque l'ennemi est soit mort, soit hors de portée. Certains loups peuvent ignorer les tours. La maison principale est en capacité d'infliger quelque dégât en derniers recours.

### **Objectif:**

Le joueur prend le camp des cochons pour défendre contre les loups.

Lorsque le joueur souhaite poser une tour, il y aura plusieurs restrictions à laquelle le joueur n'a pas le choix que de le respecter. Premièrement les tours ont une limite à laquelle il puisse poser, 5 par défaut, chaque fois cette tour est détruite elle fait augmenter cette limite de 1. Deuxièmement, le joueur doit les espacer de tel sort que les loups puissent passer, donc pas de barrage de tour. Troisièmement, les tours pourront être posé qu'avant les lancements des vagues, donc pendant les pauses entre chaque vague. Et dernièrement, si le joueur n'a pas assez d'argent pour acheter une tour, alors rien ne se passe lorsqu'il pose la tour.

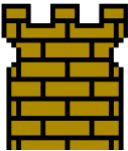
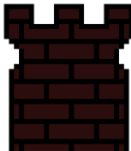

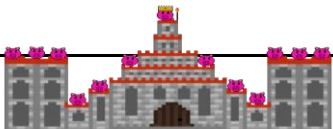
Chaque fois le joueur réussit à défendre la vague, les vagues suivantes deviendront beaucoup plus nombreux et variés, ainsi les récompenses augmentent aussi (+ 450 \* le ième vague) à chaque fin de vague, et il y a un total de 5 vagues.

Le joueur doit faire attention au point de vie restant de la maison, s'il tombe à 0 alors c'est fin de partie, et donc le joueur a perdu.




Résumé : L'objectif du joueur sera donc d'organiser la défense de la maison principale en plaçant les différents types de tours avant chaque vague. A la fin de chaque vague le joueur peut dépenser l'argent qu'il aura gagnés durant les fins de vagues, il pourra donc acheter ses tours puis les poser pour solidifier sa défense et pour se protéger contre les loups.

### Modèle :

Camp des cochons :

Tour en paille	Tour en bois	Tour en brique	Maison
			
Le moins cher et le moins résistant et fait peu de dégât que les autres tours (Tour faible)	Un peu cher et assez résistant et assez bon dégât (Tour moyen)	Très cher, très résistant et très bon dégât (Tour fort)	La cible du loup à laquelle le joueur doit le défendre, il est capable de se défendre seulement si les loups sont très proches

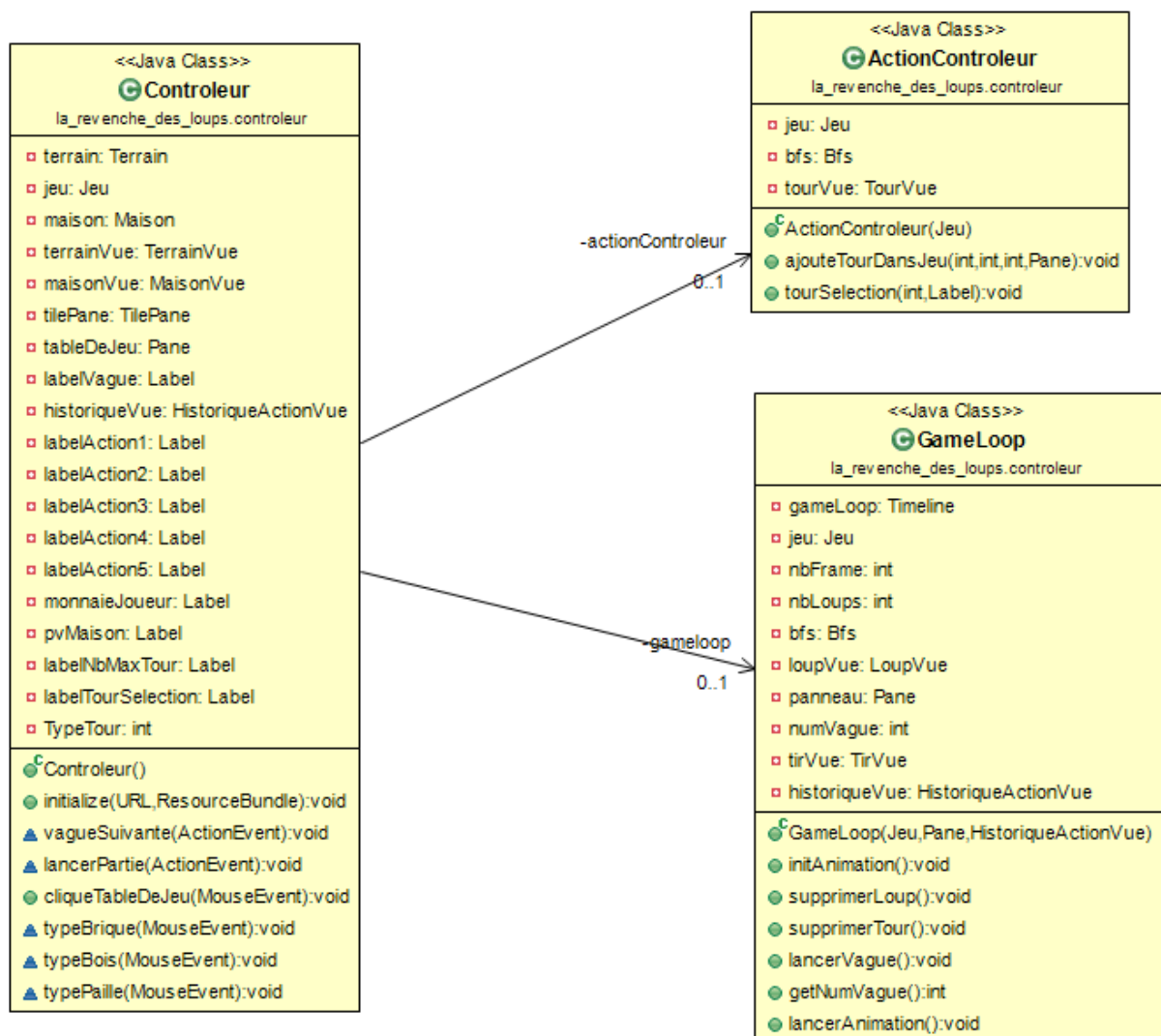
Camp des Loups :

Loup gris	Loup Blanc	Loup Fantôme
		
C'est le loup normal, qui n'a ni trait distinct ou de capacité spécial, et il est le plus fragile	Il est le plus grand et le plus costaud que les autres loups	Il fait très très mal lorsqu'il attaque, il a une probabilité d'esquiver une tour ou de les hantées (n'a pas d'influence sur les tours)

## Architecture :

Le projet est divisé en plusieurs sous packages : les packages controleur, modele, vue, mais aussi ressources test et exceptions. L'intérêt de ces différents packages est de séparer les différentes fonctionnalités du projet. Tout ce qui est en lien avec la vue, tel que les différents sprites, la Map ou encore l'affichage de l'historique d'action sont intégrés au package vue, tandis que toutes les classes en rapport avec la partie ou encore les fonctionnalités des personnages sont dans le package modèle.

Quant aux packages contrôleur, voici un diagramme de classe pour comprendre son fonctionnement :

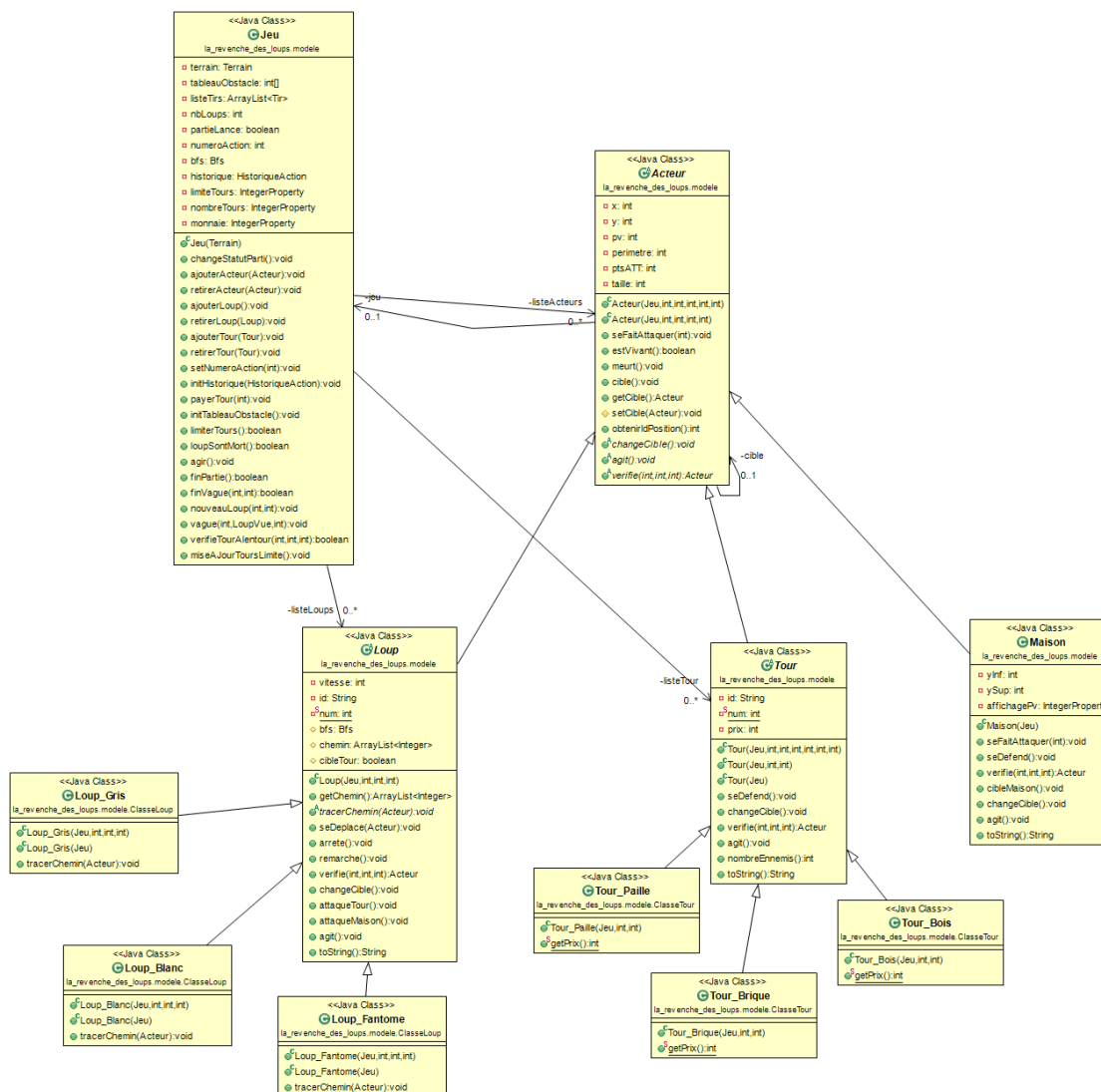


## Diagrammes de classe :

Notre modèle est constitué de 5 classes majeurs. La classe Jeu est la classe principale du modèle, c'est celle qui va lier le modèle avec le contrôleur et qui va gérer les différents événements qui ont lieu lors d'une partie. La classe Acteur est la classe mère des différents acteurs : Les loups, les tours, et la maison. On a choisi d'en faire une classe abstraite pour permettre aux sous classes d'utiliser les mêmes méthodes, mais différemment selon les acteurs.

Ces acteurs héritent donc de la classe Acteur et ont chacun des attributs similaires mais dont les valeurs sont différentes, tel que les point de vie, le périmètre d'attaque, ou encore les point d'attaques.

Comme il y a 3 types de loups et 3 types de tours, les classes Loup et Tour possèdent chacune trois sous classes : les classes Loup\_Gris, Loup\_Blanc, Loup\_Fantome pour les loups, et les classes Tour\_Paille, Tour\_Bois, Tour\_Brique pour les tours. Voici un diagramme de classe (sans les getters) des différentes classes que nous venons de décrire :



## Diagrammes de séquence :

Déroulement de la méthode changeCible

1 : Loup utilise la méthode getCible()

2 : Acteur retourne la cible de L1

3 : Loup utilise la méthode estVivant() avec la cible retourner en 2

4 : Acteur renvoi false (sinon la méthode changeCible s'arrête ici)

5 : Loup utilise la méthode cible() qui lui permet de choisir une cible

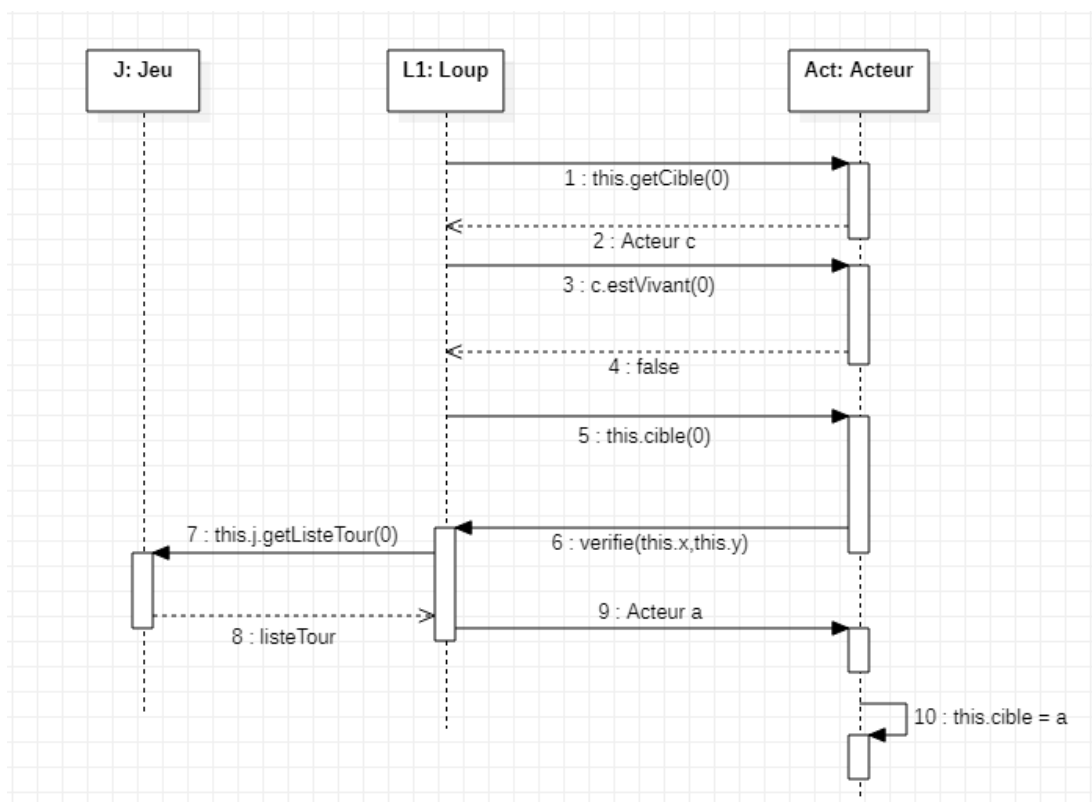
6 : Dans la méthode cible acteur utilise la méthode vérifie avec this.x et this.y en paramètres

7 : Dans la méthode vérifie() L1 appelle getListeTour()

8 : J retourne listeTour

9 : L1 finit la méthode vérifie et retourne un acteur a

10 : Dans acteur this.cible prends la valeur de a



## **Units et Exception :**

On retrouve dans notre programme un package test et un package exception.

Il y a 2 classes test : LoupTest et TourTest. Ces classes couvrent certaines méthodes des classes Loup et Tour, tel que les méthodes changeCible() ou encore seFaitAttaquer().

Dans la classe MonException se trouve une exception utilisée dans la classe LoupVue lorsqu'une image n'est pas trouvée au moment de la création d'une nouvelle Image.

## **Structure de données :**

On retrouve ici la classe jeu, acteur et bfs. Car ces trois classes représentent la majorité des données circulant à travers le jeu.

La classe Jeu : elle regroupe toutes les données, nous pouvons aussi dire que c'est le moteur principal du jeu, sans elle le jeu ne pourrait pas être déroulé comme nous le souhaitons.

La classe Acteur : elle est la superclasse de loup, tour et maison, en prenant en compte aussi les sous classes de loup (Loup Gris, Loup Fantôme et Loup Blanc) et tour (Tour Paille, Tour Bois et Tour Brique).

La classe Bfs : elle permet de générer un chemin pour un type d'acteur en particulier qui est le loup, car c'est le seul parmi les acteurs qui puisse se déplacer, elle s'appelle, Bfs car elle se comporte comme telle.



## **Utilisation maîtrisé Algorithmes intéressants :**

Nous allons voir en détail la classe Bfs:

Elle comporte trois attribut jeu de la classe Jeu, terrain de la classe Terrain pour permettre de récupérer la dimension du terrain, puis un tableau d'obstacle à laquelle elle enregistre en fonction du terrain si telle endroit un obstacle ou non.

Nous allons nous intéresser particulièrement la méthode cheminBfs, la méthode passageBfs et la méthode listeDepartArriverBfs, ces trois méthodes permettent de créer le chemin que les loups vont suivre.

passageBfs ; répertorie tous les données (récupéré dans les paramètres) dans les listes et de marquer les sommet comme utilisé tout en prenant en compte si le prochain sommet est marqué ou non.

listeDepartArriverBfs : permet de récupérer les liste de données traitées par passageBfs via les paramètres et de faire parcourir sur tout le terrain.

cheminBfs : renvoie un liste de chemin à laquelle le loup doit suivre.

Tout d'abord nous allons cloner le tableau obstacle en tableau de marquage car le loup ne doit pas déplacer dans les obstacles. La liste listeDepart et listeVersCible sont un répertoire de sommet, grâce à l'indice elle permet de connecter ces deux listes pour savoir d'où le sommet commence et d'où elle arrive. A partir de cela nous allons commencer par le point d'arriver (récupérer dans le paramètre) vers le point de départ (récupérer aussi dans le paramètre), puis on l'inverse car on ne veut pas faire déplacer depuis la maison vers le loup.