UNIVERSITÉ DE BORDEAUX

# Project POO, UBomb

Cyprien Boyer, Gilles Souton

10 janvier 2021

## 1 IMPLÉMENTATION GÉNÉRALE

### 1.1 Règles et modification

— Les monstres ne peuvent pas se marcher dessus.
— Le temps d'explosion d'une bombe est d'une seconde.
— Les monstres violet (bleu) sont "intelligents" et suivent le joueur.
— Les monstres sont plus rapides dans les derniers niveaux.

### 1.2 Game

Contient tout le jeu, les mondes, le nombre de mondes, le niveau actuel dans lequel se trouve le joueur.

#### *1.2.1 Dépendances*

Le bon fonctionnement de la classe Game ici repose sur config.properties, le fichier doit être correct, ainsi que les level(1)(2)(3).txt
On également supposé que la représentation des niveaux dans les fichiers de niveaux serait toujours des caractères d'où l'utilisation d'un BufferedReader qui est destiné à lire des caractères au lieu de bytes.
Une autre hypothèse dont tout le jeu repose dessus est que les levels sont chargés depuis des fichiers, certaines fonctionnalités ne fonctionnent donc pas si le jeu est chargé en static comme fait au début du projet.

### 1.3 Bombes

Les bombes sont considérés comme GameObjects, chaque objet représentant une bombe possède leur propre AnimationTimer, leur permettant de garder une trace du temps.

#### *1.3.1 Discussion conception*

On a voulu implémenter le design pattern : State pattern, puisque la bombe change d'état en fonction du temps, mais en prenant en considération le faible nombre d'états de la bombe on c'était un peu trop.
On aurait pu également utiliser le main update de GameEngine pour mettre a jour le timer de la bombe sans utiliser de Animation Timer.

## 1.4 Monstres

Même chose que la bombe si l'objet était plus compliqué, on aurait pu utiliser State Pattern, mais même chose les états ne sont pas si nombreux. Et exactement comme les Bombes ils possèdent leur propres AnimationTimer pour garder une trace du temps
Ce choix à été fait pour que comme les bombes ils peuvent constamment continuer a actualiser leur positions et se déplacer, même pour les niveaux non affichés. Sans nécessairement garder une référence aux objets dans GameEngine, ainis GameEngine n'a que des références de gameObject qui sont affichés a l'écran à l'exception des bombes qui sont un peu particulières. Les monstres ont également une "AI" très simple qui calcule un court chemin jusqu'au joueur, a chaque déplacement du joueur, et utilise ce chemin pour se diriger vers le joueur. Ce "smart" Monster est représenté sur une map par '@', en contrepartie tant que le joueur n'est pas dans leur monde ils ne se déplacent plus (ils attendent pratiquement le joueur).

### *1.4.1 Discussion conception*

Le fait que les monstres peuvent constamment se mettre à jour dans chaque niveau peut causer une baisse de performance si il y aurait beaucoup plus de niveaux et également beaucoup de monstres, Mais ce choix à été fait car si la bombe continue d'exploser lorsque l'on change de niveau, si les monstres s'arrêtent de bouger alors il peut devenir très facile d'exploiter cette faille pour tuer des monstres. Dans le cas où l'on aurait plus de 3 niveaux on pourrait seulement mettre a jour 1 niveau avant et après le joueur. (Exemple : le joueur est dans level2 on ne met à jour alors que les monstres dans les level1 et level3).

## 1.5 GameEngine

Comme pour Game les sprites GameObjects (Box et Monster) sont regroupés dans une liste, et les bombes sont dans une hashMap regroupant chaque bombe avec une liste de sprites. Seul les éléments du monde dans lequel le joueur se trouve est rendu a l'écran.

### *1.5.1 Discussion conception*

On aurait pu créer les Bombes dans player lorsque la requêtes de créer une bombe est reçu, et après GameEngine aurait du chercher pour l'objet dans le world associé, Ici GameEngine crée la bombe permettant de l'ajouter a la Hashmap de bombe directement et la bombe s'ajoute elle même dans le world correspondant.

# 2 NOTES

## 2.1 Javadoc

Vous pouvez utiliser gradle pour créer la JavaDoc.

## 2.2 AI

L'IA est très basique et simple, l'algorithme de plus court chemin est basé sur A*, le seul point différent est sur la performance où l'on n'a pas utilisé de tas, résultant sur un algorithme un peu moins performant. De ce fait les monstres ne cherchent que un chemin lorsque le joueur change sa position, et non pas à chaque frame. Si un monstre se retrouve alors bloqué par un autre monstre peut résulter à une destination qui n'est pas le joueur, mais qui est ensuite corrigé lorsque le joueur se déplace de nouveau.

La vitesse des monstres est également différente dans chaque niveaux, 1 case par seconde dans le premier niveau, 1 case toutes les 666ms dans le deuxième niveau et dans le dernier niveau 1 case toutes les 333 ms.
Certains points n'ont pas été abordé dans ce pdf, car on se devait de ne pas le rendre trop long.