**Département Informatique**

Diplôme préparé : D.U.T.

Projet Tutoré

Tower Défense

Gourou Jérémy, Yohan Lechat, David Ornella, Hervé Omgba, Juliana Delfaut, Bettuzzi Luc, Charles-Eric Letendart

# Tower Défense

# Le terrain

Tout d’abord, après avoir réfléchis et nous êtres questionné sur la base du programme, il nous a semblé qu’une des meilleures façons de faire était de placer une image en arrière plan pour que le jeu évolue sur cette dernière. On à pu voir que cela n’était pas difficile du tout et qu’on pourrait même faire une interface présentation du jeu proposant les options comme lancer jeu, option et quitter. On a donc choisit de se lancer dans la programmation de cette façon. L’inconvénient de cette technique est qu’il faut tout de même gérer le positionnement absolu des principaux acteurs. On a donc placé des boutons et du texte en les superposant à l’image de fond en faisant plusieurs tests. On à ensuite tracé une « grille ». Travailler avec des coordonnées n’a jamais été ce que le groupe voulait. Pour un tel jeu, on était tous d’accord pour travailler avec des tableaux et des cases. Notre terrain de jeu (la partie verte sur terrain.pcg) à donc été découpé en fonction de ces dimensions de façons à réaliser notre grille et à créer nos cases .

# Les Ecrans

Dans la suite du lancement de la programmation de notre groupe, on a gérer notre programme par Vue Ecran. On a donc créé quatre écrans, soit quatre images de font sur lesquelles sont superposé du code.

L’ « EcranPrincipal » qui est la première interface que l’utilisateur va rencontrer et qui va l’orienter vers deux autres écrans. L’« EcranOption » dans lequel l’utilisateur aura accès aux options du jeu et L’«EcranJeu » L’écran jeu représente la majeur partie du programme, c’est cette classe qui va gérer le déroulement du jeu.

Enfin l’ « EcranFin » qui permettra d’afficher le « gameover »

EcranPrincipal

EcranFIN

EcranJeu

Ecran Option

# Les vagues

Les vagues sont préparées à l’avance dans le code avec des ArrayList de Monstres. Elles sont ensuite gérer dans le déroulement du programme au niveau de « EcranJeu » à l’aide d’un compteur de vagues. Les monstres choisis pour la vague et stocké dans une arrayList vont être recopié les uns après les autres dans une arrayList « TabMonstre » de EcranJeu. Une fois le compteur de Monstre atteint, l’ArrayList est « pleine », tous les monstre ont été importé et ont pu spawn. L’opération va se répété ensuite pour toutes les vagues. La Vague nous sert aussi à savoir quand le jeu tourne de façon à incrémenter les sous.

# Les manques :

* Les graphismes : le côté graphique peut largement être amélioré. Le dessins.png pourrait être bien plus beau.

On pourrait imaginer changer la map au bout de 10 vagues et repartir pour 10 de plus sans trop de difficulté. Il faudrait juste changer le terrain.png en coloriants des carrés au milieu de la map en imaginant que c’est un ou plusieurs trous. Les cases concernées auront le booléen « vide » à false et nôtre Dijkstrat sera toujours opérationnel.

Les tours focus bien un monstre, met on en voit pas les tires. Il faudrait rajouter un dessin de tir venant de la position de la tour et allant jusqu’à la position du monstre.

* Les ressources : ce que l’on à appeler le mana et qui doit servir à recharger les tours et qui peut donc être assimilé à des munitions n’a pas été gérer.
* Le Dijkstra de nôtre projet : Quelque soit le point de départ (sur toute la ligne gauche) l’algorithme va comparer tous les chemins possible afin de sauvegarder et d’utiliser le chemin le plus court. Cependant un des principes utilisé pour le créer est « toujours aller de l’avant ». Le monstre ne sait pas reculer. Du coup, si le défenseur fait une sorte de C à l’envers, le monstre va s’engouffrer dedans sans pouvoir en sortir.
* Les objets Tours : Il aurait fallu rajouter d’autre tour et gérer plus leurs évolutions
* La partie réseau : Nous n’avons pas gérer le réseau et nous n’avons donc pas laissé la possibilité de jouer à plusieurs machines.

# Répartition des tâches

Terrain : Luc

Déplacement : Jérémy

Tours : Yohan

Interface : Juliana

Gestion Or, incôme, Point de vie, Mana : Charlie

Monstres : David

Chef de projet : Hervé (+ aide interface)

EcranPrincipal/EcranOption/EcranFin : Charlie Yohan David

EcranJeu : partie faite avec tout le groupe où l’ont à du liée tout le travail réalisé.

# Les monstres

Les monstres sont défini dans le packtage modèle, on a une classe monstre abstraite qui sert de base à nos différents monstres que nous allons présenter ultérieurement. Les monstres ont tous en commun certains attributs :

\_ des points de vie « pointdevie » : entier

\_ un prix  « prix » : entier

\_ une capacité d’attaque « dmg » : entier

\_ un cout « cout » : entier

\_ une rente d’or « income » : entier

\_ un booléan ciblable pour savoir si le monstre peut se faire tirer dessus « Ciblable » : booleen

\_ une clée « key » : entier

Le monstre de base se construit avec ses coordonnées sur le terrain : « IndexCaseH » et « IndexCaseL » qui permette de cibler une case du jeu.

## Monstre basique

Un monstre basique est un monstre qui étend la classe monstre, il va donc hériter de tous les attributs de la classe monstre et on va en plus fixer son « cout » et son « prix » à 10. Ses « dmg » à 10 et ses « pointdevie » à 100 .

## Monstre puissant

Un monstre puissant est un monstre qui étend la classe monstre, il va donc hériter de tous les attributs de la classe monstre et on va en plus fixer son « cout » à 50, son « prix » à 100. Ses « dmg » à 30 et ses « pointdevie » à 200 .

## Monstre volant

Un monstre volant est un monstre qui étend la classe monstre, il va donc hériter de tous les attributs de la classe monstre et on va en plus fixer son « cout » à 100, son « prix » à 250. Ses « dmg » à 80 et ses « pointdevie » à 200 .

# Les Tours

Les tours sont défini dans le packtage modèle, on une classe abstraite qui sert de base pour créer les différentes tours que l’on va utiliser dans le jeu. Les tours ont donc en commun certains attributs :

\_ des points de vie : « point de vie » : entier

\_ un prix « prix » : entier

\_ des dégâts d’attaque « dmg » : entier

\_ un gain d’or récurrent « Income » : entier

\_ des coordonnées « IndexCaseH » « IndexCaseL » : entier

\_ un booléen pour savoir si la tour attaque « AttaqueEnCour » : booleen

\_ une cible « cible » : entier

\_ un niveau « level » : entier

\_ une portée d’attaque « portee » : entier

\_ un sprite « sp » et une Texture « Text » permettant de modéliser visuellement la tour

## Tour1

On est allé au plus simple, la tour1 possède 10 « pointdevie » son « prix » vaut 100, ses dégâts « dmg » sont de 50, son « Income » 1 et sa portée « 2 »

## Tour 2

Cette tour possède 10 « pointdevie », son « prix » est de 350, ses dégâts « dmg » sont de 100, son « Income » 1 et sa portée « 2 .