Projet PJI 110 : Tower Defense

Etudiants et Auteurs : CHARNEUX Dimitri et LEPRETRE Rémy
Encadrant : ROUTIER Jean-Christophe

Sujet : Création d'un jeu de Tower Defense en 2 Dimension en Java

INFORMATIQUE 2016

SOMMAIRE

Remerciements

Avant de commencer notre rapport, nous tenons à remercier les personnes qui ont contribué à la réussite de notre projet.

Tout d'abord, nous souhaitons remercier toute l'équipe pédagogique de l'Université de LILLE1 pour leurs enseignements pendant ces quatre années qui nous amener jusqu'à la fin de ce Master.

Enfin, nous tenons à remercier tout particulièrement Monsieur ROUTIER Jean-Christophe pour nous avoir suivi tout au long de la réalisation de ce projet et pour l'aide qu'il nous a apporté.

Introduction

Lors de notre première année de Master INFORMATIQUE, nous avons eu l'opportunité de réaliser un projet en autonomie, qui nous tenait à cœur.

Nous avons choisis de réaliser un jeu de type Tower Defense basé sur celui déjà existant Dungeon Defender II.

Un Tower Defense consiste à défendre un certain point, que nous appellerons ici le nexus, contre plusieurs vagues d'ennemis qui avancent pour tenter de le détruire.

Pour défendre ce nexus, le joueur a la possibilité d'incarner un personnage et de poser des défenses afin d'empêcher les vagues d'ennemis d'atteindre le nexus.

Pour cela, nous avons décidé de réaliser ce projet en JAVA en utilisant la librairie Swing. Mais contrairement au jeu d'origine, le nôtre sera réalisé en deux dimensions.

Dans un premier temps, nous allons vous présenter plus en détail notre projet. Ensuite, nous détaillerons les principaux points du développement de notre jeu. Puis, nous expliquerons les différentes possibilités d'évolutions et d'améliorations de ce Tower Defense. Enfin, nous terminerons par une conclusion de cette expérience.

I/ Présentation du projet

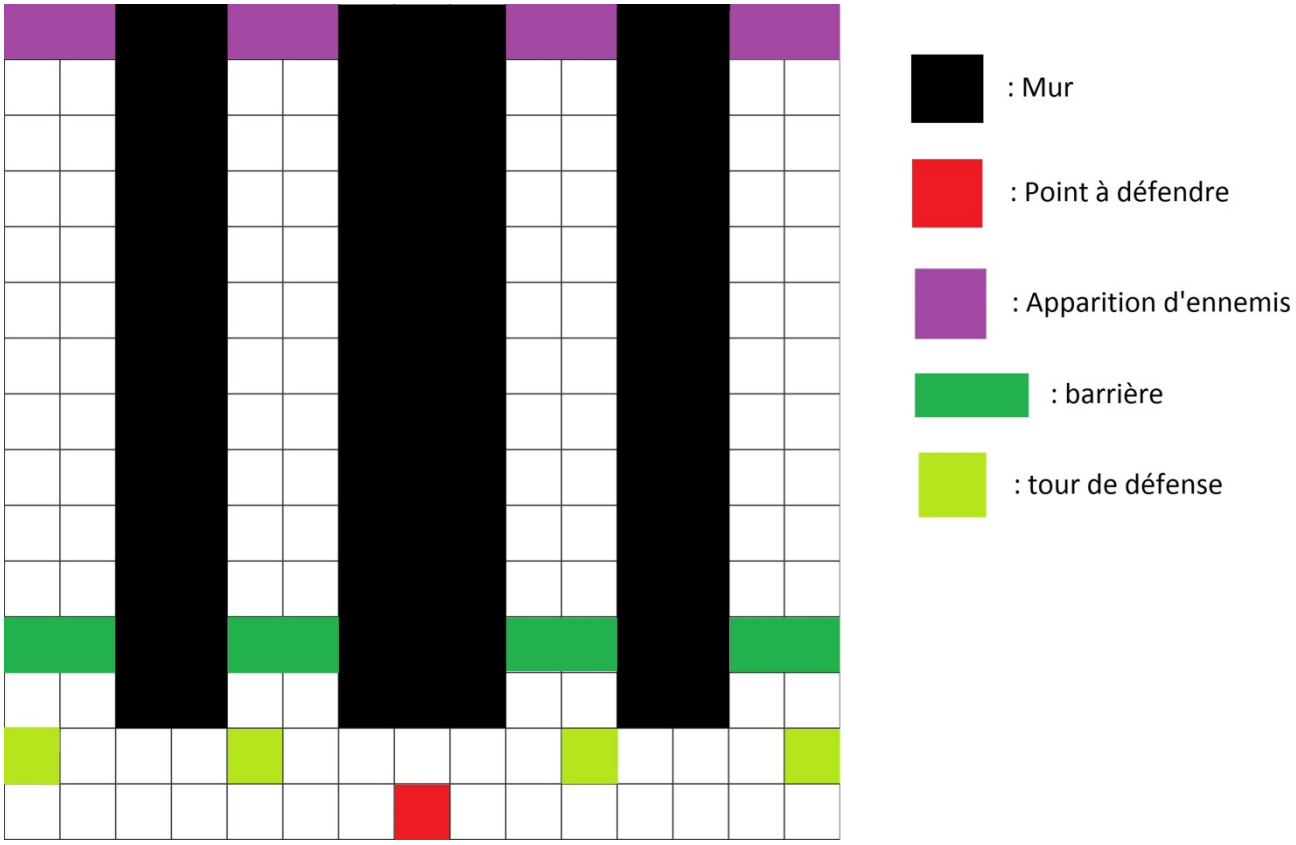
Nous avons décidé pour ce PJI de créer un jeu de type Tower Defense. Pour vous expliquer ce projet, nous allons tout d'abord vous montrer ce qu'est un Tower Defense avant de se concentrer sur les spécificités de notre jeu.

Un Tower Defense est un jeu dont le but est de défendre un point, le *nexus*, face à plusieurs vagues d'ennemis. Pour ce faire, le joueur a la possibilité de poser des défenses pour bloquer l'avancée adverse. Les défenses du joueur peuvent être de plusieurs types, des barrières pour simplement ralentir la progression ennemis ou des tours qui pourront les attaquer afin de les détruire. Le joueur peut utiliser son personnage pour attaquer directement les adversaires.

Une partie s'arrête quand le nexus a été détruit ou si le joueur a réussi à détruire tous les ennemis en gardant le nexus en vie.

Le jeu se déroule sur plusieurs cartes qui ont les mêmes caractéristiques :

- le ***nexus*** que le joueur doit à tout prix défendre
- des ***murs*** pour délimiter la zone
- les ***chemins*** que les vagues ennemis vont emprunter.



Vous pouvez voir ci-dessous le schéma type d'une carte de notre Tower Defense.

Il y a plusieurs types d'ennemi avec des caractéristiques propres :

-les ***points de vies*** qui indiquent leur état de santé
-une ***puissance*** qui indique les dégâts occasionnés à chaque coup
-une ***portée*** qui indique à quelle distance un monstre peut toucher sa cible
-une ***vitesse*** de déplacement.

Chaque type d'ennemi a un comportement propre. Il peut vouloir avancer vers le nexus sans se préoccuper des autres défenses, il peut attaquer chaque défense à sa portée ou cibler ses attaques sur un certain type de défense.

Le personnage incarné par le joueur possède lui aussi les mêmes caractéristiques.

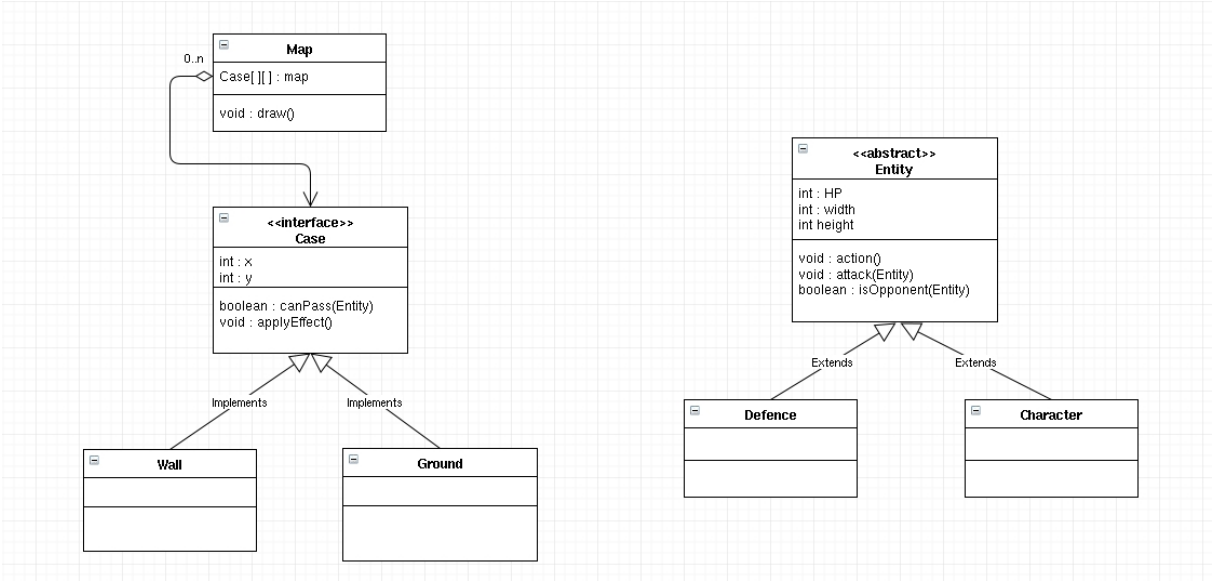
Concernant les défenses, elles auront les mêmes caractéristiques excepté la vitesse qui sera forcé à 0 car elles ne se déplacent pas.

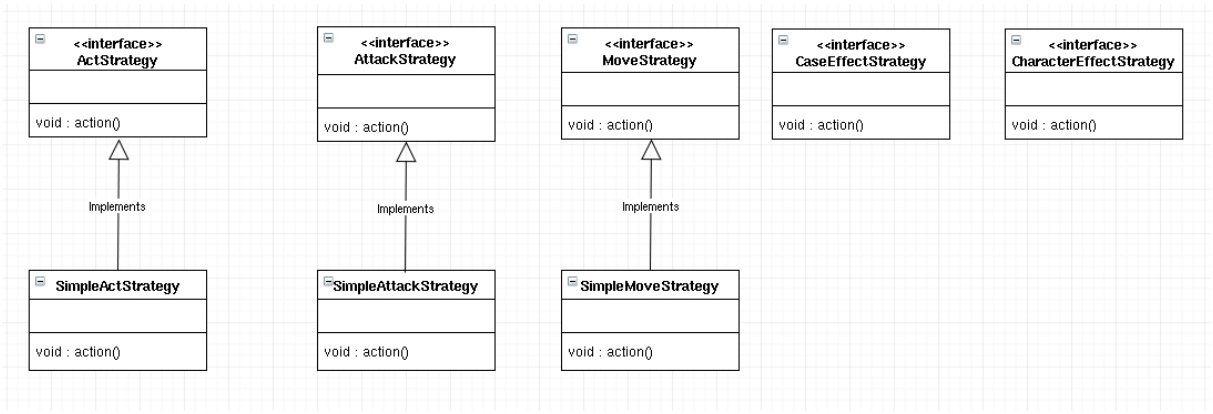
Dans ce texte, nous appellerons « entités » l'ensemble comprenant les ennemis, le personnage du joueur ainsi que les défenses.

Le jeu se déroule en plusieurs ***tours***. Un tour est une unité de temps durant laquelle chaque entité pourra exécuter une action. Nous avons choisit d'appliquer les dégâts causés par une entité à la fin de chaque tour pour éviter les déséquilibre et permettre à toutes les entité d'attaquer.

II/ Travail réalisé

a. UML





b. Développement

Dans cette partie, nous allons aborder les points principaux du développement de l'application dans l'ordre où ils ont été réalisés.

1.Personnages

Nous avons tout d'abord commencé par le développement des différents personnages qui peuvent intervenir dans le jeu :

- les ***Characters***, qui regroupent d'un côté les différents personnages que le

joueur va pouvoir incarner et de l'autre les monstres que celui-ci devra affronter.
- les ***Defences*** que le joueur pourra poser dans le jeu et dont le nexus, ne pouvant pas attaquer ni être placé par le joueur, fait partie.

Plusieurs raisons nous ont incités à créer deux classes différentes pour les Characters et les Defences. Tout d'abord, une case peut contenir une liste de Characters et une seule Defence. En optant pour une seule classe, nous aurions pu mettre une Defence dans la liste de Characters ou un Character comme une Defence ce qui n'est pas souhaitable dans notre jeu. Ensuite, Defences et Characters ont une différence au niveau du déplacement, une Defence est immobile tandis qu'un Character peut se déplacer, l'utilisation des deux classes permet de prendre en compte ce point.

Nous avons également défini qu'un personnage incarné par le joueur pouvait passer à travers ses propres défenses mais que les monstres eux ne le peuvent pas.

La gestion du personnage et la possibilité de poser des défenses sont les deux derniers composants qui ont été ajouté aux entités.

Ces deux composantes sont primordiales pour le jeu, la première va permettre au joueur de manipuler un personnage pour attaquer les troupes ennemies. Le joueur pourra ainsi déplacer son personnage avec les touches Z Q S D ou les flèches directionnelles en fonction de ses préférences. Il pourra aussi attaquer en appuyant sur la touche espace.

Ensuite, pour poser ses défenses afin de protéger le nexus, le joueur devra positionner le curseur de la souris sur la case où il souhaite placer sa défense, puis effectuer un clic gauche pour la poser. Pour apporter plus de stratégie au jeu, nous allons instaurer un système de pièces. A chaque ennemi tué, le joueur gagnera des pièces, celles-ci pourront être utilisées pour poser les défenses. Le joueur devra donc faire attention à son nombre de pièces pour pouvoir poser des défenses au bon moment.

2. Stratégie

Puis nous sommes passés au développement de l'intelligence artificiel des entités. Pour ce faire, nous avons décomposé le comportement des entités en ***stratégies*** qui vont gérer les actions exécutée par l'entité. Il existe trois types de stratégie pour régir cela :

- les ***moveStrategy*** qui sont les stratégies utilisés par les entités afin de se déplacer (ex : avancer tout droit), le déplacement d'une entité est basé sur une de ses caractéristiques qui est la vitesse couplé à une variable globale qui est une distance parcouru en un tour en pixel.
- les ***attackStrategy*** qui sont les stratégies que vont employer les monstres pour attaquer (ex : attaquer l'ennemi le plus proche). Pour la détection des monstres à attaquer, cela dépend de la caractéristique Range de chaque Character qui est une distance en pixels. La map possède une liste de tous les personnages et cette liste est parcourue afin de déterminer la distance entre le monstre attaquant et le monstre de la liste en cours de parcours afin de savoir s'il est à portée d'attaque ou non.
- et enfin les ***actStrategy***, qui sont le déroulement principal de l'IA d'un monstre (ex : aller directement jusqu'au nexus pour le détruire). Les actStrategy analysent l'environnement (ennemis, alliés, chemins) de l'entité qui leur est associée et choisissent si l'entité doit attaquer ou se déplacer. Elles appellent ensuite la méthode action de l'attackStrategy ou de la moveStrategy pour que l'entité effectue l'action souhaitée.

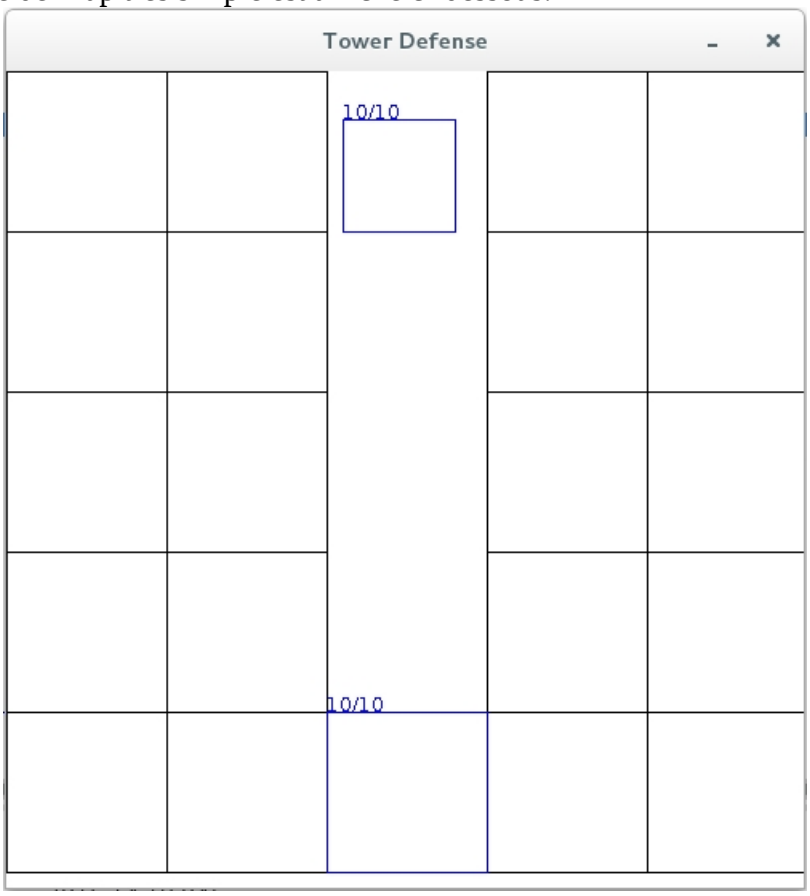
Pour définir l'IA d'un monstre il faut donc lui définir une moveStrategy, lui définir une

attackStrategy et enfin dans une actStrategy définir la corrélation qui existe entre attaque et move.

3. Carte

Ensuite nous sommes passés au développement de la Map. Elle représente le terrain de jeu sur lequel va s'effectuer la partie et sera composée d'un nombre de cases de même taille préalablement défini par le niveau en cours. Il y a les cases ***« sol »*** où les Characters peuvent se déplacer et sur lequel le joueur peut placer des défenses, une défense prend au minimum une case et on ne peut poser qu'une seule défense par case, et les cases ***« mur »*** à travers lesquels aucun personnage ne peut passer et sur lesquels il est impossible de poser des défenses.

Nous avons réalisé une interface Graphique où un mur est représenté par un carré au bord noir et une entité par un carré au bord bleu au dessus duquel est affiché les points de vie de l'entité. Un exemple de Map très simple est affiché ci-dessous.



Nous avons ensuite voulu simplifier la création de ces Maps. Pour cela, nous avons utilisé des fichiers dans lequel une map est détaillée. Ces fichiers respectent un format et contiennent les cases présentes dans la carte. Un mur est représenté par le symbole « X », un case « sol » par le symbole « o » et le nexus par le symbole « N ».Au lancement d'un niveau, le programme va donc lire le fichier associé et créer la carte avant de lancer la partie.

c. Choix

Nous avons également effectué quelques choix mineurs mais nécessaires à préciser :

- un Character peut être sur plusieurs cases (1, 2 ou 4)
- un Character attaque depuis le centre de son image mais peut être touché n'importe où sur celle-ci
- pour la map, les coordonnées horizontales sont les x et verticales les y. Le point en haut à gauche de la fenêtre est de coordonnée (0,0)

III/ Perspectives d'évolutions

Pour parvenir au bout de notre projet, il reste plusieurs choses à faire. Nous allons voir les différents éléments qui pourront être ajouté ou modifier à notre projet dans le futur.

a. Diversifier

Le premier éléments à ajouter à notre jeu sera des nouvelles cartes. Nous allons créer différentes cartes réparties par niveaux. Chaque niveaux abordera un thème et aura des graphismes associé à celui-ci.

Nous pensons également à intégrer un éditeur de niveaux qui va permettre aux joueurs de créer ses propres cartes et les partager avec d'autres joueurs.

Nous voulons créer également d'autres ennemis ainsi que d'autres défenses. Pour ce faire, nous allons ajouter d'autres stratégies d'attaques, de déplacements et d'actions. Nous pourrons ainsi ajouter des ennemis attaquant en priorité les défenses et d'autres attaquant le joueur humain. Nous pensons également à faire de nouvelles défenses qui cibleront les ennemis ayant peu de points de vie pour éclaircir un peu le champ de bataille. D'autres défenses et adversaires seront ajoutés au fur et à mesure que nous les imaginerons.

b. Attaque et équipement

Un autre éléments à ajouter serait un système d'équipement. Les équipements portés par le joueurs modifieront ses statistiques pour lui permettre de venir à bout de niveaux plus difficiles.

Ces équipement pourront être achetés dans une boutique en échange d'or récupérer en terminant un niveau. Il pourront également être lâché par des monstres que le joueur aura vaincu.

Nous envisageons également à ajouter d'autres attaques pour le personnages contrôlé par le joueur. Il pourra alors réaliser des attaques de zone ou des attaque lui permettant de récupérer de la vie. Chacune de ces attaques coûtera au joueurs des points de ***mana*** qui se rechargeront tout seul au cours du temps.

c. Effets

Les attaques lancées par les entités pourront avoir des effets. Il existera plusieurs types d'effets appliqués aux attaques :

- Le ***poison*** et le ***feu*** infligent des dégâts à chaque tour à l'ennemi touché
- Le ***froid*** peut ralentir les ennemis durant un laps de temps et le gel peut même les immobiliser.
- L'effet ***confusion*** peut quant à lui retourner un ennemi contre les siens pendant quelques secondes.

Ces effets peuvent aussi être appliqués à des cases, ainsi, une case enflammée brûlera les ennemis se déplaçant dessus. Les effets appliqués à des cases sront actifs pendant un nombre prédéfini de tours.

Enfin, certaines entités qu'on appelle « entités de soutien » peuvent soigner ou booster temporairement leurs alliés. Ainsi, une entité peut voir son attaque, sa portée ou sa vitesse améliorée grâce à un allié situé à proximité.

Pour réaliser les effets, nous allons créer des stratégies de la même manière que pour les attaques. Une case contiendra alors une liste d'***effectStrategy*** qui seront appliquées à chaque fin de tour.

Conclusion