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

Master 1 Semestre 2 INFORMATIQUE 2016

SOMMAIRE

Remerciements

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

Tout d'abord, nous souhaitons à remercier toutes l'équipe pédagogique de l'Université de LILLE 1 pour leur enseignement pendant ces quatre années qui nous amener jusqu'au Master.

Enfin, nous tenons à remercier tout particulièrement Monsieur ROUTIER Jean-Christophe pour nous avoir suivis 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 seras 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'amélioration de ce Tower Defense. Enfin, nous terminerons par une conclusion de cette expérience.

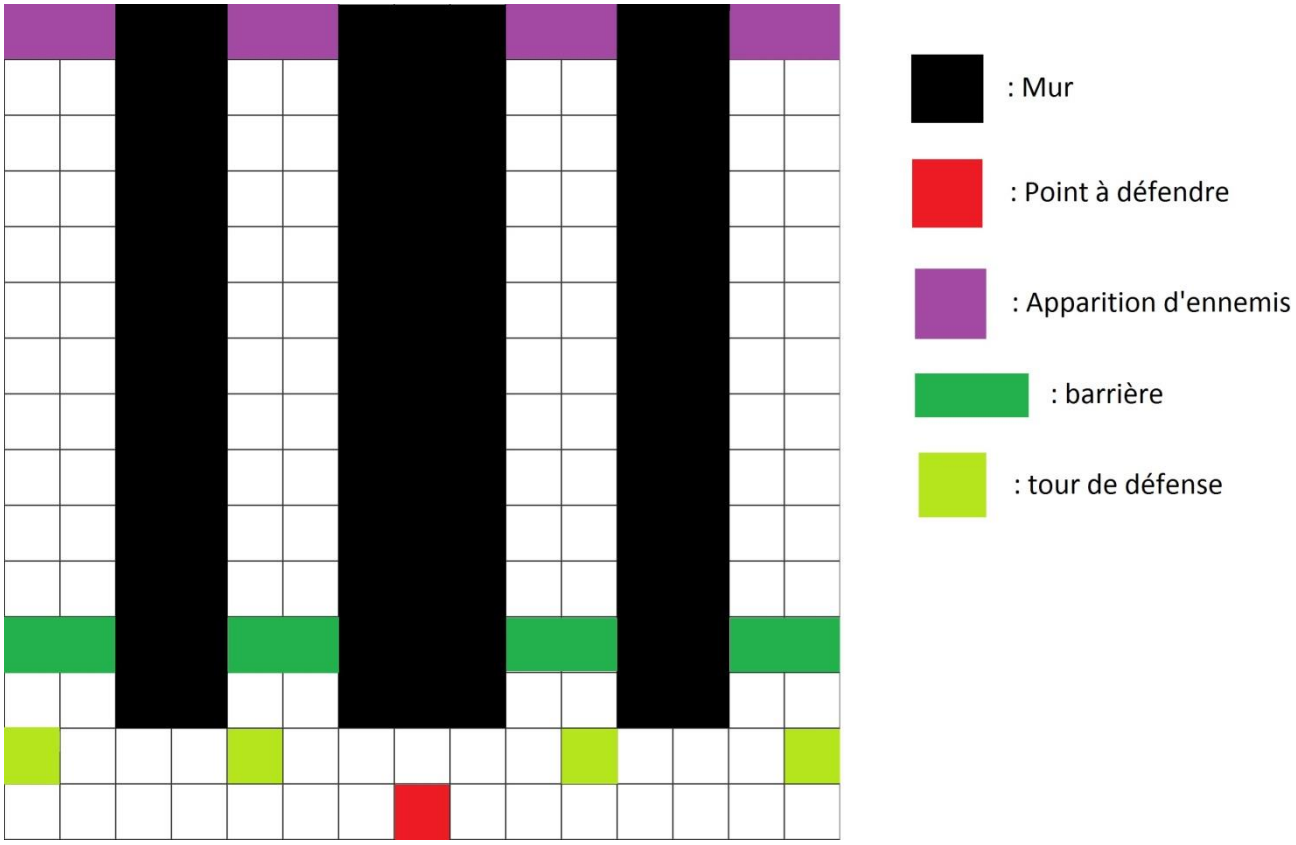
I/ Présentation du projet

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 intact.

Le jeu se déroule sur plusieurs cartes qui ont les mêmes caractéristiques : un point que le joueur doit à tout prix défendre, des murs pour délimiter la zone et 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 attaque qui indique les dégâts fait à chaque coup sur les défenses, une portée qui indique à quelle distance un monstre peut toucher sa cible et une vitesse de déplacement. Chaque type d'ennemi a un comportement propre. Il peut vouloir avancer vers le nexus sans ce préoccupé des autres défenses, il peut attaquer chaque défenses à sa portée ou ciblé 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 car elles ne se déplacent pas.

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

Les attaques lancées par les entités peuvent avoir des effets. Il existe plusieurs types d'effet appliqué aux attaques :

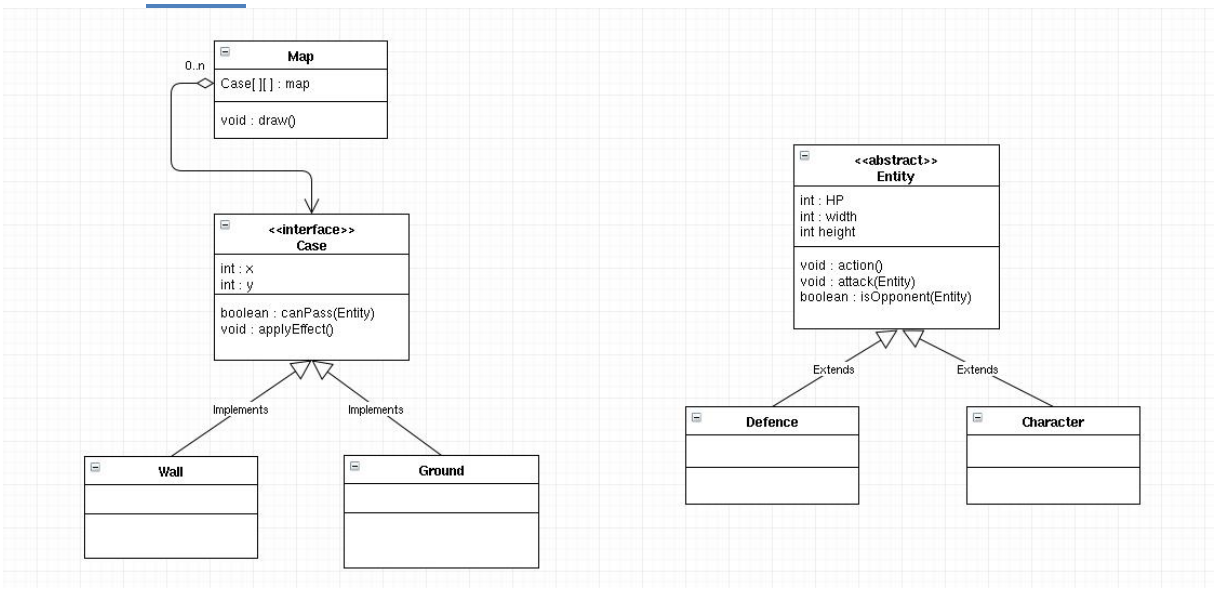
- Le poison et le feu infligent des dégâts à chaque tour à l'ennemi touché
- Le froid peut ralentir les ennemis durant un laps de temps et le gel peut même les immobiliser.
- L'effet confusion peut quant à lui retourner un ennemi contre les siens pendant quelques secondes.

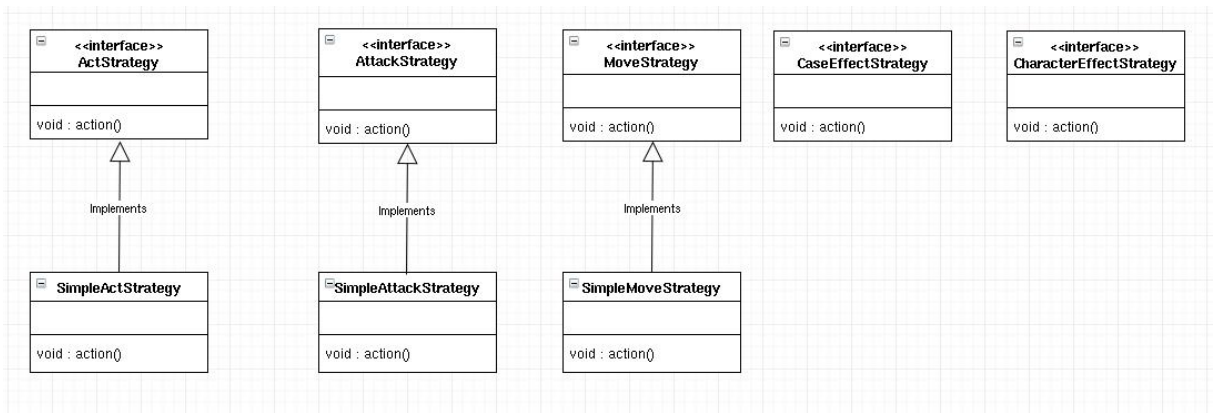
Ces effets peuvent aussi être appliqués à des cases, ainsi, une case enflammée brûlera les ennemis se déplaçant dessus.

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

II/ Travail réalisé

a. UML





b. Développement

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

- les Characters, qui regroupe 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 en échange de monnaie dans le jeu et dont le nexus (point à défendre), ne pouvant pas attaquer ni être placé par le joueur fait partit.

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 incarner par le joueur pouvait passer à travers ses propres défenses mais que les monstres eux ne pouvaient pas.

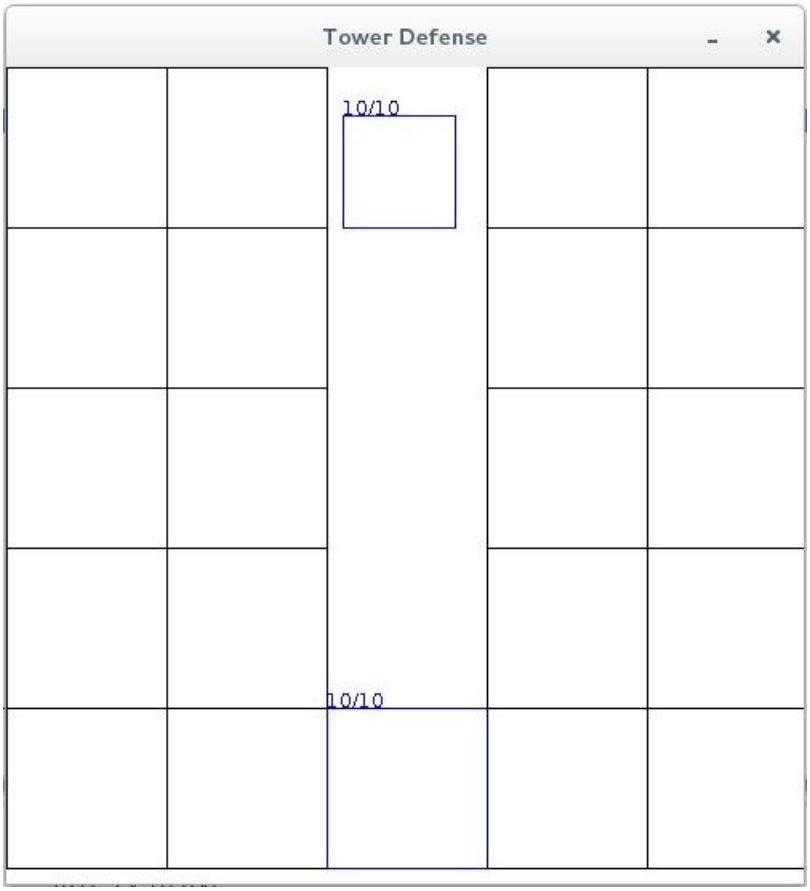
Ensuite nous sommes passés au développement de la Map. Celle-ci est composée d'un nombre de cases préalablement défini par le niveau en cours et dont les cases comportent toutes la même taille. Il y a les cases sols où les tous les Characters peuvent se déplacer et sur lequel le joueur peut placer des défenses, une défense prend au minimum une case et on ne peut poser qu'une seule défense par case, et les cases murs à travers lesquels aucun personnage ne peut passer et sur lesquels il est impossible de poser des défenses.

Puis nous sommes passés au développement des différentes stratégies (ou IA) qui vont être utilisés par les montres. Il existe trois types de stratégie pour régir cela :

- les moveStrategy qui sont les stratégies utilisés par les monstres afin de se déplacer (ex : avancer tout droit), le déplacement d'un monstre est basé sur une de ses caractéristique qui est speed couplé à une variable globale qui est une distance 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 pixel. 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 ou non d'attaque.
- 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, allié, chemin) de l'entité qui leur est associé et choisissent, en fonction de celui-ci, 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.

Enfin nous avons réalisé une interface Graphique sur lequel apparait des murs de chaque côté et un chemin au milieu sur lequel un monstre avance directement jusqu'au nexus pour le détruire.



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 d'abord voir les choses que nous souhaitons terminer avant la fin du semestre puis les choses que nous voulons faire si le temps le permet, ou, si il ne le permet pas, qui pourront être ajouté plus tard à notre jeu.

Nous allons tout d'abord implémenter deux éléments dans le jeu : la gestion du personnage et la possibilité de poser des défenses. 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. Par la suite, nous pourrons envisager une attaque secondaire avec une autre touche pour apporter de la diversité au jeu.

Ensuite, nous allons permettre au joueur de poser ses défenses pour protéger le nexus. Il 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 pièce. 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 si la partie tourne mal.

Après avoir implémenté ces deux éléments primordiaux, nous voulons, pour faciliter la création de cartes, pouvoir créer des niveaux à partir de fichiers textes ou XML. Ces fichiers respecterons un format et contiendrons les cases présentes dans la carte, ainsi que les ennemis qui vont apparaître dans le niveau. Au lancement d'un niveau, le programme va donc lire le fichier associé et créer la carte avant de lancer la partie.

Par la suite, pour apporter de la variété au jeu, nous voulons créer 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.

Nous voulons également améliorer l'aspect graphique du jeu en ajoutant des images pour les personnages, les défenses, ainsi que pour les cartes.

La dernière chose que nous allons ajouter pendant ce semestre sont les effets. Nous voulons que les entités présentes sur le terrain puissent utiliser des effets. Ces effets seront de plusieurs sortes : tout d'abord des effets négatifs comme le feu ou le poison qui infligeront des dégâts à un ennemi, ou d'autres pouvant faire baisser leurs statistiques comme la vitesse, la puissance ou la portée. Ensuite, nous allons inclure des effets positifs comme le soin ou des améliorations de statistiques.

Un effet pourra être utilisé de deux façons : soit en l'appliquant à un ennemi, celui-ci se verra appliquer l'effet pendant un nombre de tour propre à chaque effets, soit en l'appliquant sur une case. De cette manière, chaque entité marchant sur la case subira l'effet appliqué.

Enfin, pour continuer ce projet après le semestre, nous avons d'autres éléments qui pourront enrichir le jeu. Nous pourrons ajouter un éditeur de niveaux pour permettre de les créer plus facilement, ajouter un système d'équipement pour le personnage contrôlait par le joueur, ajouter des attaques spéciales pour ajouter plus de stratégies à ce jeu et ajouter un système de gestion des touches pour permettre aux joueurs d'adapter le jeu à leurs envies.

Il nous reste donc pas mal d'éléments à ajouter à notre jeu pour le compléter et lui ajouter plus de profondeur.

Conclusion