BOUALAPHA Sousada
IZACKOVIC Flavien

Programmation et algorithmique C++ – Synthèse d'Image I
RAPPORT : IMAC TOWER DEFENSE

Repo Git : https://github.com/MarechalKop/Projet-Tower-Defense-

Description des fonctionnalités implémentées

L'application est un jeu de type tower defense où le joueur doit défendre sa base contre des vagues successives d'ennemis en plaçant stratégiquement des tours de défense. L'objectif principal est d'empêcher les ennemis d'atteindre le dernier nœud du parcours en utilisant les tours pour les éliminer.

Règles du jeu :

Le jeu consiste en trois vagues d'ennemis successives. Le point de départ des ennemis sur la carte se situe au niveau d'une case bleue tandis que le point d'arrivée à défendre est représenté par une case rouge. Le joueur place des tours le long du chemin des ennemis. Chaque tour peut attaquer les ennemis à portée à l'aide de boulets de canon.

Il existe deux types d'ennemis, chacun avec ses propres caractéristiques :

Type 1 :
- Points de vie : 50
- Vitesse : 80
- Récompense : 20

Type 2 :
- Points de vie : 550
- Vitesse : 50
- Récompense : 50

Le jeu propose deux types de tours, chacune ayant aussi des caractéristiques distinctes :

Type A :
- Puissance : 20
- Portée : 3 (en distance de Chebyshev)
- Cadence de tir : 1 (en dixième de seconde)
- Prix : 100

Type B :

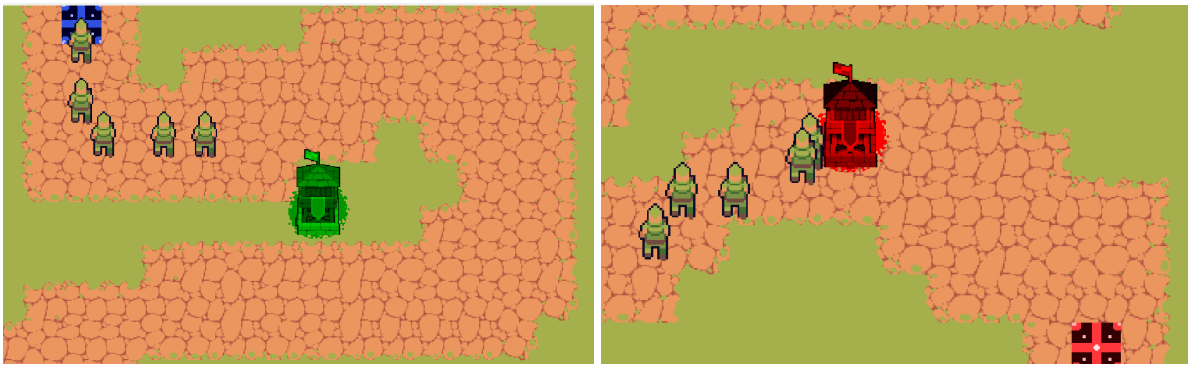
- Puissance : 80
- Portée : 7
- Cadence de tir : 40
- Prix : 200

Le jeu contient un système d'argent : au cours du temps, le joueur se voit recevoir une petite somme d'argent (récompense) et chaque ennemi éliminé rapporte de l'argent au joueur ce qui lui permet d'acheter des tours. Le joueur gagne la partie s'il élimine toutes les vagues d'ennemis. Le joueur perd la partie s'il perd ses 3 vies : il en perd une à chaque fois qu'un ennemi atteint le point d'arrivée.

Guide d'utilisation :

Notre jeu se joue uniquement avec les touches du clavier, nous avons fait ce choix pour des raisons techniques : il nous a paru bien plus difficile d'implémenter un système prenant en compte la position sur la fenêtre de jeu que d'assigner des commandes spécifiques pour certaines touches du clavier.

Pour commencer une partie, il suffit au joueur d'appuyer sur la touche P.
La première vague commence alors, le joueur commençant sa partie avec 100 pièces, il peut donc acheter la tour à 100 pièces en appuyant sur A.
La tour qu'il a choisie apparaît maintenant sur la carte du jeu. Le joueur peut maintenant choisir l'endroit où la tour sera construite en la déplaçant avec les flèches du clavier. Si celle-ci peut être placée à cet endroit, elle apparaît en vert. Dans le cas contraire, elle est de couleur rouge.



Si la tour est bien placée, il suffit au joueur d'appuyer de nouveau sur A pour placer sa tour.
Dans le cas où le joueur veut poser une tour à 200 pièces, il lui suffit de répéter la même opération en appuyant sur la touche Z au lieu de A.
Enfin, si le joueur veut quitter le jeu, il peut appuyer sur la touche Q.

Post Mortem

Analyse de notre travail :

Nous sommes plutôt satisfaits de ce que nous avons réussi à produire dans le temps imparti, nous sommes conscients que c'est véritablement perfectible mais à notre hauteur le produit final est convenable. Alors qu'il s'agit de notre premier "gros" projet en C++, nous avons réussi à : partager le travail idéalement, le segmenter en une multitude de petites tâches individuelles bien plus simples à appréhender, satisfaire la majorité des points du cahier des charges, gérer OpenGL.

Nous avons rencontré de nombreux problèmes au cours du projet. Le premier a été la création d'un fichier CMake permettant d'utiliser en même temps la bibliothèque OpenGL et la bibliothèque sil permettant de lire les images. Ce problème a été résolu en demandant de l'aide à des camarades plus expérimentés que nous. Le second gros problème qui nous a demandé beaucoup d'effort a été l'implémentation des tours dans le jeu : il a fallu d'abord utiliser tout le travail fait sur les ennemis et sur les graphes, créer les projectiles, essayer de visualiser leurs déplacements dans la console, puis enfin mettre en place la sélection et le placement des tours sur la carte du jeu. Globalement, ce qui nous a le plus ralenti à été le sentiment d'avancer à l'aveugle, souvent sans résultat visible sur la fenêtre du jeu. Mais nous avons donc appris à ajouter des vérifications sous forme de std::cout pour nous en sortir.

Avec plus de temps, on aurait un peu plus soigné notre code en le répartissant dans plus de fichiers .cpp et .hpp afin d'éviter le spaghetti code que l'on retrouve dans le fichier main.cpp. Mais notre plus grand regret est de ne pas avoir un peu plus soigné l'aspect graphique de notre projet. Dans ce domaine, tout reste à faire mais l'implémentation de texture est si longue et fastidieuse que nous avons décidé de le laisser de côté pour se concentrer sur les fonctions essentielles de notre projet. En bref, on aurait aimé avoir une interface plus propre, des animations pour les tours et les ennemis et une carte plus détaillée avec des éléments de décor comme des fleurs, des cailloux, des panneaux, des buissons, etc.

Sources des assets

Bouton "QUIT" : https://pixelartmaker.com/art/ab924b085d1c780
Pack de sprites utilisés pour les cases de la carte, les tours et les ennemis :
https://craftpix.net/sets/tower-defense-top-down-pixel-art/
Image du coeur : https://i.ytimg.com/vi/3KWS-8Ljog4/maxresdefault.jpg
Image de la pièce :
https://encrypted-tbn0.gstatic.com/images?q=tbn:ANd9GcS0dnph1C2hdGc2aFmXnC8eOPf0GZ33c_YrRA&s
Image de boulet :
https://static.wikia.nocookie.net/dungeon-masters/images/e/e4/CannonBall.png/revision/latest?cb=20210806031111