Projet C++

IDE utilisée, dépendance et lancement du programme :

Le projet a été réalisé sous Visual Studio 2017 et nécessite d'être compilé en configuration debug, x86. Le projet nécessite 3 dépendance GLFW, glut32 et OpenGL. Le projet peux être configurer en x64 à la sortie de git, pour pouvoir compiler il faut donc repasser en x86 si c'est le cas.

Choix de conception :

Le but du projet étant de réaliser un Tower défense les choix de conception suivant ont été effectué :

1. Point de vue utilisateur :

Du point de vue de l'utilisateur le jeu se déroule sous forme de vagues. La vague actuelle est afficher sur le côté supérieur droit de l'écran de jeu. L'argent et la vie se trouvent sur le côté supérieur gauche de l'écran. Les vagues se lancent automatiquement.

Pour placer une tour il suffit de cliquer sur la case désirée. Un fois le clic réalisé, des fenêtres contextuelles vous permettent de choisir quelle tour doit être placé. Pour placer une tour vous devez être en possession d'assez d'argent. Si vous n'êtes pas en possession de la somme requise la fenêtre contextuelle apparaîtra en rouge.

Trois tours sont disponibles :

- les tours de base dites « Cheap » coutent peu chers et on des attributs plutôt équilibrés.
- les tours « BigShot » qui tire plus lentement mais plus fort.
- les tours « SplitShot » qui tire 3 tire simultanément mais qui inflige moins de dégâts.

Un fois vos tours placés vous avez la possibilité d'améliorer leurs caractéristiques, cliquer sur la tour fait apparaitre de nouvelles fenêtres contextuelles. Vous pouvez améliorer les dégâts et la cadence de tir de vos tours.

Le jeu se termine lorsque la vie du joueur atteint 0 ou lorsque celui-ci atteint la vagues 15.

2. Code :

Deux classes compose l'architecture principales du projet :

- GameBoard elle regroupe les éléments relatifs au gameplay. C'est cette classe qui contient les cases du Tower Defense. Celles-ci contiennent les tourelles.
  Un type générique TowerBase généralise les tours et leur interface.
  La classe Player centralise les informations relatives au joueur (Argent et vie).
  Le GameBoard conserve les informations à la case sélectionné par le joueur.
- UIObjects qui centralise les objets relatifs à l'interface utilisateur. Ceux-ci sont repartie en 2 catégorie, les objets interactifs et les objets non interactifs.

Pour permettre un affichage évolutif des classes héritant de « DrawMethod » ont été créés.