Compte rendu de projet
Projet C++ / QT : le MePorg

Lilian Cizeron et Simon Provot
Cybersécurité du Logiciel - 1ère année

Table des matières

Compte rendu de TP
Le MePorg

Année universitaire 2020 - 2021

Lilian Cizeron et Simon Provot

Introduction :

Dans le cadre de notre première année à l'ENSIBS en Cybersécurité du Logiciel, nous avons une initiation au langage C++, un langage de programmation compilé orienté objet, parmi les plus utilisés dans le développement de logiciel. Afin de découvrir ce langage, nous allons programmer un petit jeu vidéo, à l'aide de la librairie QT, librairie implémentant des fonctions pratiques pour concevoir un jeu vidéo.

Le but de ce projet est bien de nous faire découvrir la mécanique du langage C++, notamment au travers des classes et objets, de la gestion de la mémoire ou bien des exceptions. Nous ne traiterons donc pas ici l'aspect sécurisation de notre logiciel.

Notre projet :

Nous avons choisi de partir sur un petit jeu au tour par tour, mélangeant le style du célèbre jeu *Pokémon*, et celui de *For The King*. Nous allons donc créer un jeu qui permet d'effectuer des duels en un contre un, qui s'enchaine. Le but du jeu est donc de réussir à tuer le maximum d'ennemis avant de mourir.

Pour ce faire, le joueur commence par choisir le type de son héros. Il choisit ensuite un nom et un métier. Le type et le métier du héros sont très important, car ils font varier ses points de caractéristiques.

Une fois ces choix effectués, les combats peuvent débuter. Pendant le combat, le joueur dispose de quatre capacités, propres au type de son héros. Il doit en sélectionner une puis l'ennemie attaque. Une fois que l'ennemi est mort, un autre arrive, et ce jusqu'à ce que le joueur meure ou qu'il n'y a plus d'ennemis à combattre. En effet, il y a un nombre aléatoire d'ennemis au lancement du jeu, afin d'éviter qu'il y ai un scénario de combat qui permette à coup sûr de gagner.

Le joueur dispose également d'un sac à dos, lui permettant de stocker des objets obtenus lorsqu'un ennemi meure. Ces objets permettent d'augmenter certaines caractéristiques du héros.

Notre projet embarquera de l'audio, nous utiliserons donc la version 5.12 de QT, afin de disposer du plugin *multimedia*.

Année universitaire 2020 - 2021 Lilian Cizeron et Simon Provot

Le diagramme de classe :

Pour bien commencer le projet, nous avons établi un diagramme de classe sous *Papyrus* afin de clarifier la structure du projet :

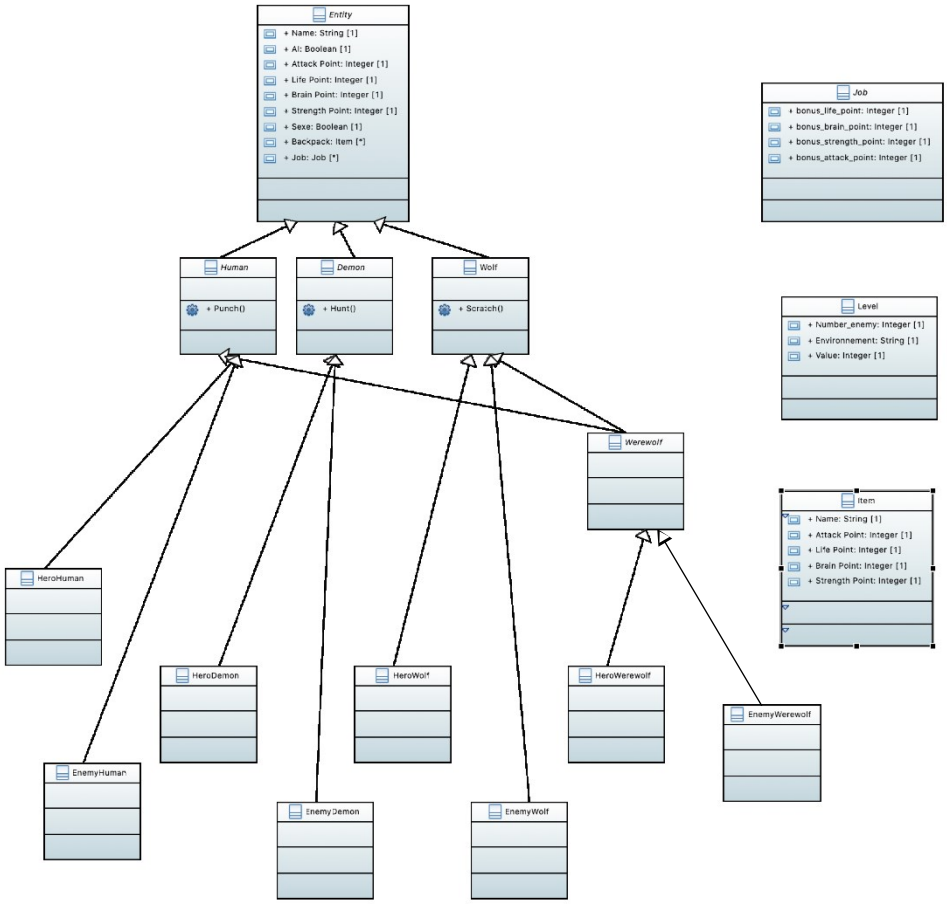

Figure 1 : diagramme de classe du projet

Sur ce diagramme, on retrouve plusieurs classes abstraites. On voit également que tous les ennemis et héros seront de type *Human, Demon, Wolf* ou *Werewolf* qui héritent eux-mêmes de la classe abstraite *Entity*. On peut observer que la classe abstraite *Werewolf* hérite de deux classes, *Human* et *Wolf*. De plus, on retrouve trois autres classes, *Job, Level* et *Item*.

Ce diagramme de classes est bien évidemment non exclusif. Nous allons être amené à effectuer des modifications légères qui ne seront pas structurelles.

La programmation :

Pour programmer notre jeu vidéo, nous avons utilisé l'environnement de développement *QT Creator*, l'IDE natif de *QT*. Ensuite, nous nous sommes réparti le travail. Simon se chargera de la partie logique du programme, et Lilian de la partie affichage du projet.

Nous avons également initialisé un dépôt git sur la plateforme *Forgens* de l'Université, afin de facilité la mise à la disposition du projet pour l'enseignant et pour nous deux. La mise en place de ce dépôt à nécessité une prise de contact avec la *Direction des Services d'Informations* de l'Université, car la plateforme n'autorisait que des envois inférieurs à 5 Mo, dû au faible espace de stockage de disponible. Disposant de fichier audio *.wav* de plus de 5 Mo, nous ne pouvions pas envoyer notre travailler sur le serveur, et c'est pourquoi nous avons donc échangé avec la *DSI* afin d'augmenter cette limite à 50 Mo.

Nous utilisons également un client git, *GitHub Desktop*, afin de faciliter les transferts sur le serveur.

La majorité du code s'enchaine grâce aux signaux des animations. En effet, lors d'une attaque par exemple, chaque phase de l'attaque dépend de l'animation qui lui est associé. Ainsi, un ennemi ne pourra attaquer le héros que lorsque celui-ci aura fini son animation d'attaque, qui est exécuté en parallèle de la partie logique de son attaque. C'est donc le signal émis par l'animation de l'attaque du héros qui va déclencher l'attaque de l'ennemi.

Nous ne détaillerons pas le code dans ce rapport, car celui-ci est commenté et très volumineux. Cela ne serait donc pas pratique de le commenter ici. Nous en parlerons également lors de notre soutenance, en expliquant notamment les choix des widgets ou la structure du programme.

Le respect des consignes :

Nous avions évidemment quelques consignes à respecter pour ce projet, voici comment nous y avons répondu :

Contrainte de l'énoncé	Exemple de réponse
Classes	L'ensemble du projet
Héritage Multiple	*Werewolf* qui hérite de *Wolf* et *Human*
Dynamic_cast	Vérification du type du héros dans *MainWindow*
Surdéfinition	Opérateur + entre une entité et un item, dans *Entity*
Exception	En cas d'échec de la lecture du fichier JSON, dans *Reader*
STL	Vecteur d'entités qui stock les ennemis, dans *Game* et *MainWindow*

Les points d'améliorations :

Notre projet est désormais pleinement fonctionnel. Il peut être utilisé pour jouer et atteindre le meilleur score possible. Plusieurs points d'améliorations peuvent néanmoins être soulevé. Tous d'abord, lorsque le programme est exécuté à travers l'IDE, une erreur de mémoire survient à la fermeture du programme, et celui-ci est interrompu. Cela ne survient qu'à la fin de l'exécution du programme, ce qui n'est pas dérangeant pour l'utiliser. Nous laissons également la main à QT et lui faisons entièrement confiance quant à la gestion de nos pointeurs, ce qui mériterait une vérification approfondi, afin d'éviter notamment les fuites de mémoires. Ensuite, nous pourrions implémenter une histoire en plus du mode de combat continu. Enfin, nous pourrions rendre l'interface interactive, de sorte que les éléments sur la page dépendent de la taille de la fenêtre et ne soient pas à des coordonnées fixes.

Conclusion :

Pour conclure, nous pouvons dire que nous avons pu apprendre, au travers de ce projet, les bases du langage C++, avec notamment son système d'héritage de classe ou de gestion de mémoire. Malgré les quelques points d'améliorations que nous avons soulevés précédemment, nous avons pu produire un petit jeu vidéo fonctionnel, dynamique, dans le temps imparti.