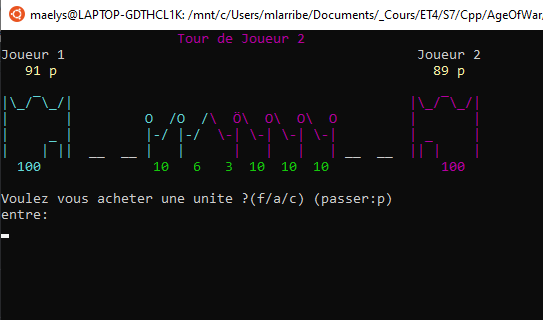
LARRIBE Maëlys 26/01/2021

**Rapport de Projet de C++**

1. **Implémentation Projet**

Ce projet a pour but de concevoir et implémenter un mini-jeu sur console. Ce jeu est inspiré de Age of War. Nous développerons ici une adaptation de ce jeu avec un mode "humain" contre "humain" et un mode "humain" contre "IA", en tour par tour.

****

Mon objectif initial était de produire un code un minimum adaptable. Cette intention se retrouve dans quelques méthodes se basant sur la généricité ou les fonctions anonymes (Terrain::add ou Terrain::Action3).

Dans ce projet toutes les unités héritent d’une classe ***Unite***. L’aire de jeu quant à elle est représentée par la classe ***Terrain,*** qui elle-même contient des conteneurs (classe ***Case***) de un pointeur d’***Unite*** chacun.

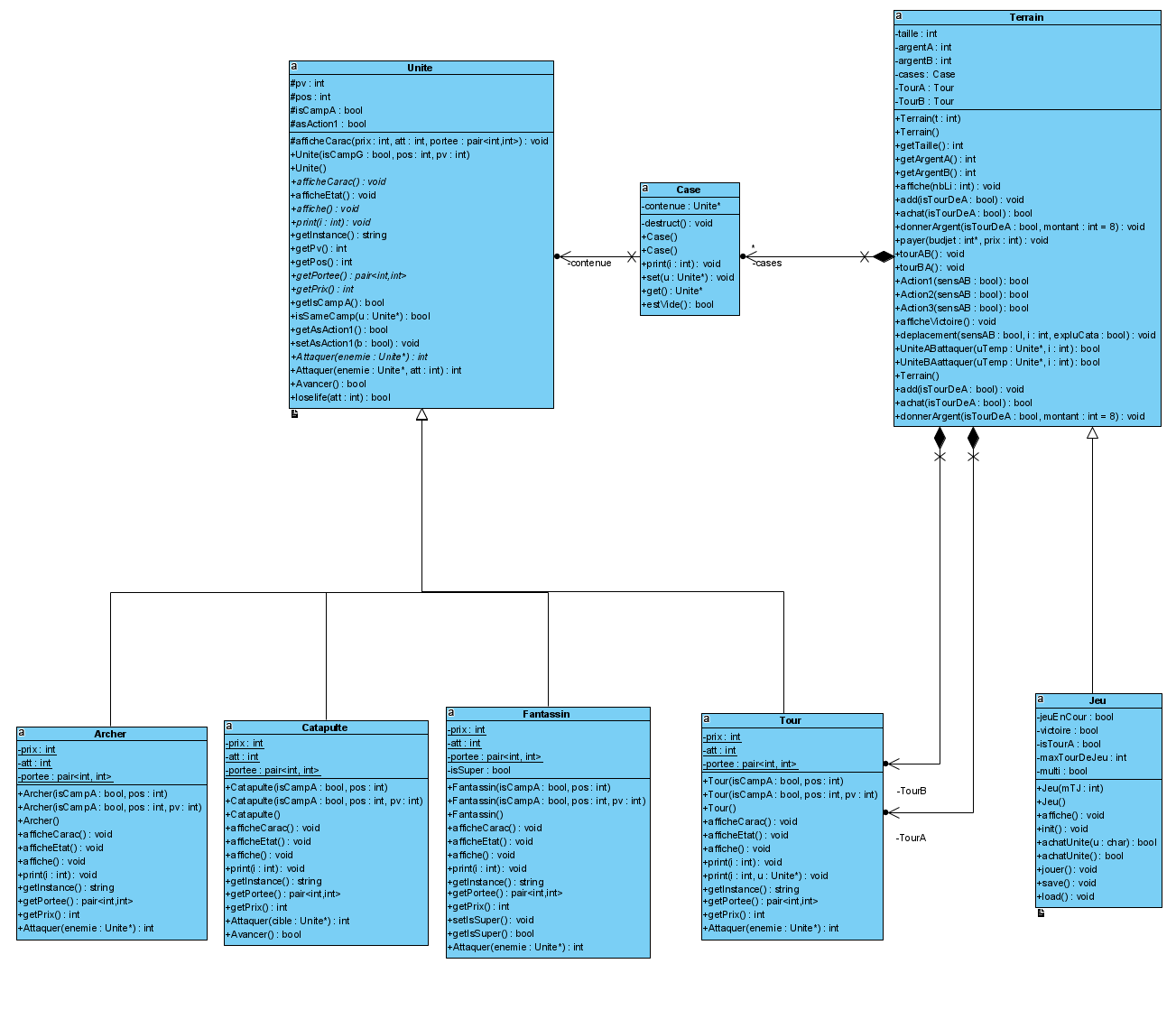
***Terrain*** référence également les deux bases.

Le jeu est exécuté avec la méthode jeu() de la classe ***Jeu.***

L’affichages des unités est fait avec des « bonhommes bâton ».

1. **Diagramme UML**

Ci-joint le diagramme d’UML de mon projet à ce jour.

****

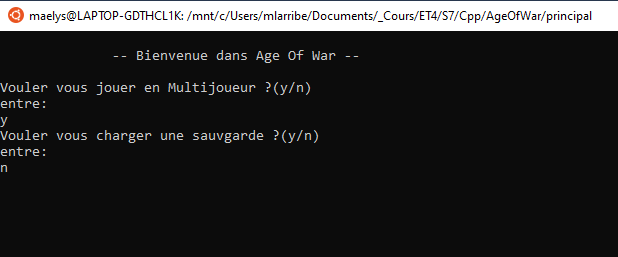
1. **Manuel d’utilisation**

Une fois avoir « make » dans le répertoire « /principal », il faut pour lancer le programme écrire « ./main ».

Tel qu’implémenté, « q » est toujours disponible pour quitter le programme.

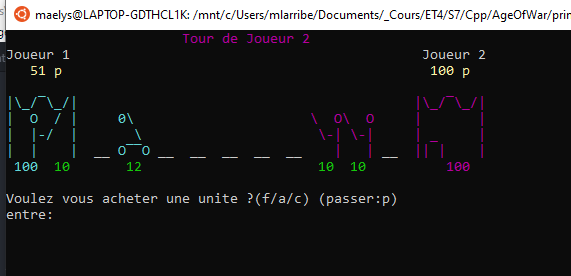
Au démarrage il est demandé à l’utilisateur s’il souhaite jouer en multijoueur ou contre l’ordinateur. (oui : « y », non : « n »).

Suite à cela, l’utilisateur à l’opportunité de charger une sauvegarde s’il le souhaite (oui : « y », non : « n »).



Une fois la partie lancée, chaque joueur est libre de placer une unité selon son argent disponible (fantassin : « f », archer : « a », catapulte : « c »).

Il a également l’opportunité de passer son tour (« p »).



Une fois la partie terminé le joueur a l’occasion de là sauvegarder s’il l’a interrompu (oui : « y », non : « n »).

1. **Difficultés et améliorations**

De nombreuses difficultés se sont présentés lors de ce projet.

**Switch case impossible :**

Pour une raison que je n’ai pas réussi à déterminé, je me suis retrouvée dans l’incapacité d’effectuer un switch case.

**Mort Tour :**

J’apprécie peu devoir faire une comparaison pour chaque Unite.

**Sur vérification :**

Certaines vérifications de pointeur non nul sont effectués à plusieurs endroits dans mon programme, elles ne sont pas nécessairement toutes utiles même si elles protègent le code.

**Historique Terrain :**

J’aurais aimé implémenter une système d’historique dans le Terrain qui stock les actions (achat) effectuer à un instant t. cela aiderais dans l’isolation de bug.

**Super-soldat :**

Pour une adaptabilité plus simple du code, j’aurais préféré prendre le temps de faire une classe à part pour super-soldat.