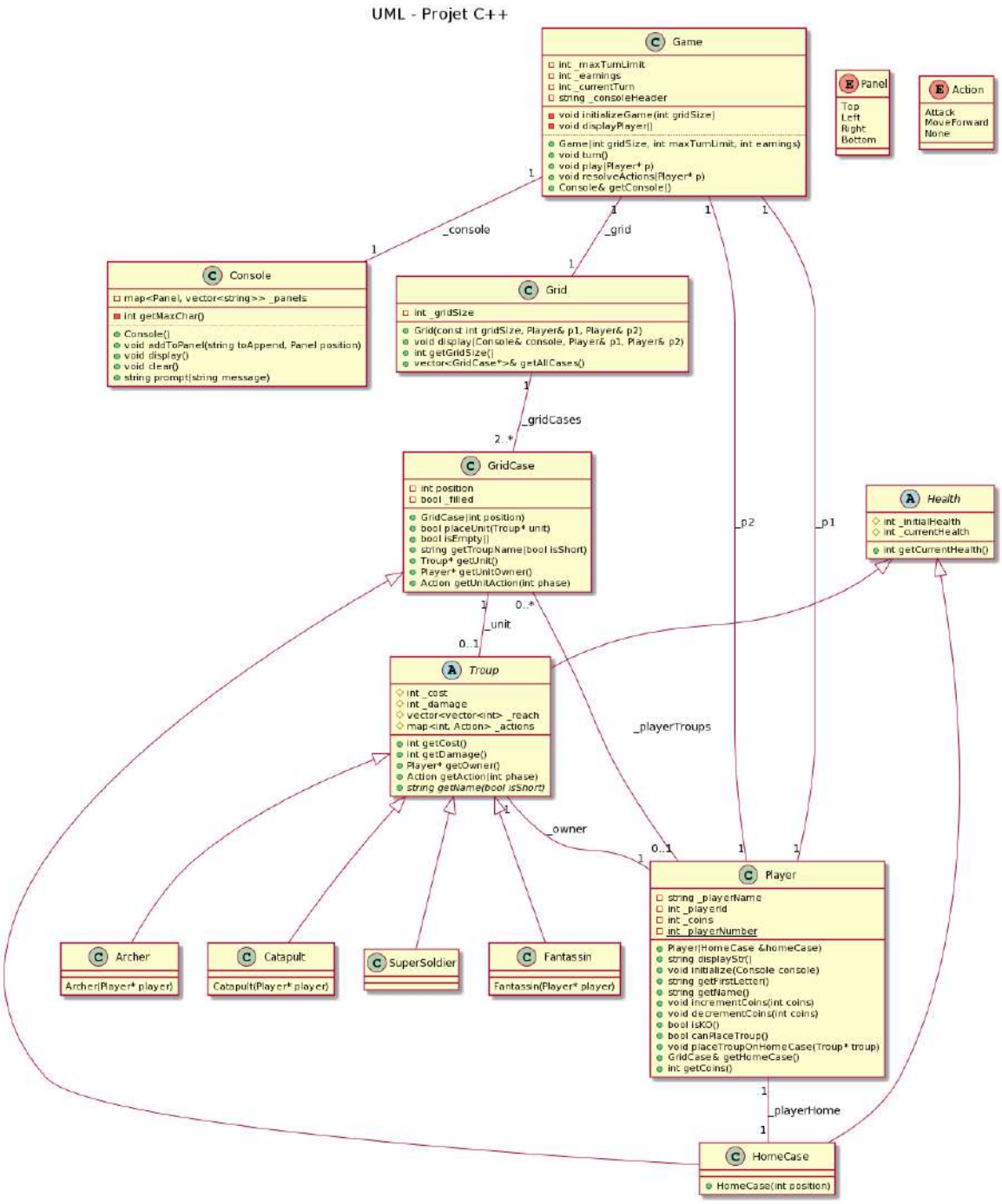
Compte rendu de projet C++

Binôme : Lucas B., Antonin D.

Choix de conception

Afin de répondre au cahier des charges, nous avons réalisé le diagramme de classe suivant :



Réalisations

Lucas B. – Structure du code (UML), Réalisation du Makefile, Affichage du jeu, Gestion des inputs utilisateur - @0xWryth

Antonin D. – Documentation, Gestion des inputs utilisateur, Gestion des troupes, Gestion des phases de résolutions du jeu - @adepreis

Classes

Game

La classe Game permet de stocker les données principales de la partie : les Joueurs, le plateau de jeu, la Console – qui nous permettra d'afficher l'état du jeu.

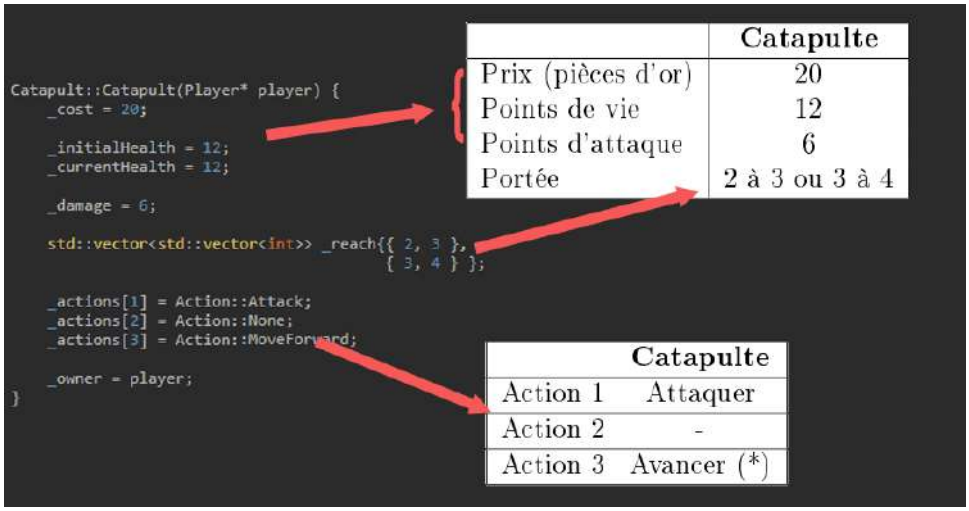
Player

La classe « Player », nous permet de manipuler et d'accéder à un certain nombre de données concernant un joueur. Les pièces possédées par le joueur y sont modifiables. L'accès au nom du joueur et de son initiale nous permet de l'afficher sur le plateau. De plus, cette classe nous permet d'accéder à la case représentant la base du joueur – nommée HomeCase – qui nous permet de faire apparaitre les troupes sur cette dernière si le joueur dispose d'une quantité suffisante d'or et si cette dernière est vide.

La classe « Health », représente les données de vie qui seront associés à la base d'un joueur et aux troupes de ce dernier.

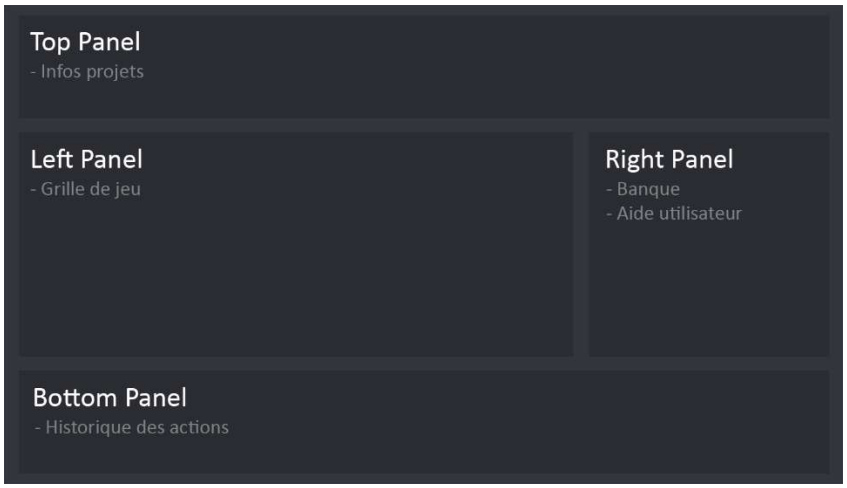
Troup

Les différents types d'unités héritent d'une classe Troup. Cela permet de mettre en commun le montant, les dommages ainsi que les points de vie (cf. classe Health) de chaque troupe. Petite particularité : les actions et la portée des attaques sont stockés de la manière suivante :



Console

Pour l'affichage, nous avons décidé de créer une classe Console afin de décomposer l'affichage de l'invite de commande en différentes sections.



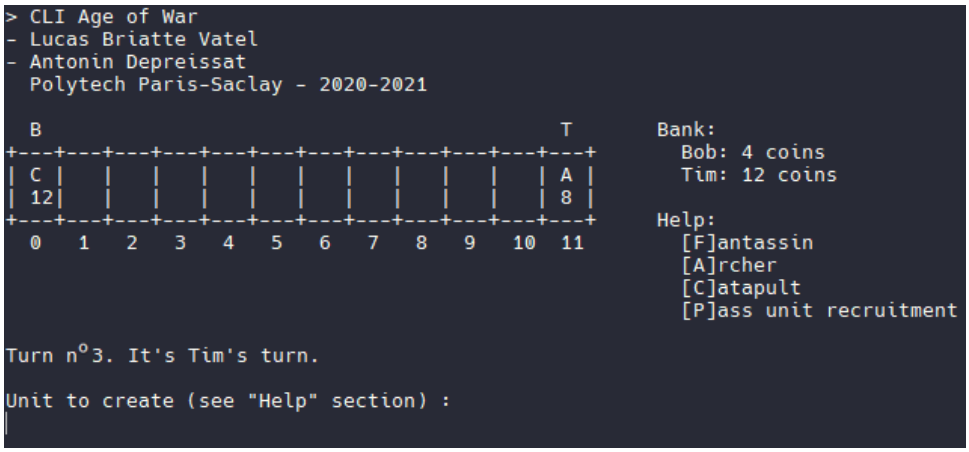
Nous nous retrouvons donc avec un affichage séparé en 4 « panneaux » indépendants les uns des autres permettant de garder de manière uniforme l'affichage.

Avancement du projet

Actuellement, les fonctionnalités suivantes sont opérationnelles :

- Player vs Player
- Mécanisme de tour-par-tour
- Représentation de l'aire de jeu (cases, bases des joueurs)
- Différenciation des types d'unités
- Affichage des points de vie restants pour chaque unité
- Affichage des pièces d'or disponibles pour chaque joueur

À l'heure actuelle, cela permet de démarrer une partie, entrer les noms des joueurs, distribuer les pièces d'or à chaque tour et acheter une première unité.



Il reste donc encore quelques parties à implémenter :

- Les phases de résolution d'actions 1, 2 et 3 ainsi que l'affichage des mouvements/attaques de chaque unité dans l'ordre
- La différenciation entre les unités des joueurs (attribution et affichage d'une couleur pour chaque joueur par exemple)
- Proposer de défier une IA (et mettre en place sa stratégie)
- Sauvegarder l'état courant d'une partie dans un fichier (pour ce faire, nous aurions implémenté des fonctions *load()* et *save()* dans la classe Game)

Difficultés rencontrées

Lors de la réalisation du projet, nous avons rencontrés différents problèmes liés à l'inclusion des classes entre elles. Nous avions plusieurs classes qui dépendaient l'une de l'autre et qui produisait lors de la compilation un souci de « dépendances circulaires » que nous avons dû résoudre patiemment.

Même si nous ne sommes pas encore allés jusqu'à la transformation de Fantassin en Super-soldat, la conception du SuperSolider est complexe. En effet, dans notre diagramme UML, c'est une spécification de la classe Troup mais nous ne savons pas encore s'il est plus judicieux que ce soit une spécification de Fantassin ou même un simple booléen, au niveau des attributs du Fantassin, indiquant sa promotion.

Actuellement, la résolution des phases d'actions n'est pas terminée et bloque un peu notre avancement. Nous faisons par exemple face à la problématique de la prise en compte d'une Action 1 est non effectuée : sous quelle forme stocker cette information, à quel niveau ?..

Bilan

Bien que le projet ne soit pas encore totalement finalisé lors de l'écriture de ce rapport, sa structure est effectivement finalisée. Même si elle a peu évolué depuis le début, certains aspects de la conception nous avaient échappé. Si nous devions le refaire, nous consacrerions plus de temps à la réflexion quant aux différents aspects de conceptions afin d'inclure davantage de concepts objets comme vu dans le cours.

La réalisation de ce projet nous a permis d'apprendre à résoudre des dépendances circulaires et à davantage comprendre l'élaboration d'un Makefile, ce qui nous aura été utile car nous programmons chacun sur des environnements différents : Windows (Lucas) et Linux (Antonin).