LO21 - P22 Projet Carcassonne

Rapport 1

DU Sylvain - TAVERNY Kelyan - BOUZAR Massil - OUEDRAOGO Taoufiq - THIBAULT Ambroise (Auteur responsable de ce rapport)

Introduction

Le présent document relate l'ensemble du travail effectué à la date du 9 avril 2022 pour le projet Carcassonne qui s'inscrit dans l'UV LO21 : Programmation et Conception orientée objet, enseignée à l'Université de Technologie de Compiègne durant le semestre d'automne 2022.

Durant ces premières semaines, nous avons analysé en détail les règles du jeu, nous avons réalisé l'importance majeure des informations concernant les tuiles que ce soient concernant leurs spécificités (attributs) mais surtout la manière dont sera contenue l'information de leur positionnement sur le plateau les unes par rapport aux autres. En effet, c'est à partir de cela que nous pourrons déterminer si une tuile a le droit d'être posée ou non et c'est également grâce à cela que le calcul du score des joueurs pourra être effectué. C'est sur cette base que nous avons mené une première analyse que nous continuerons à pousser au fur et à mesure de l'avancée de notre projet.

Nous avons également longtemps réfléchi à l'implémentation des extensions du jeu. Une attention particulière a été porté à cet aspect du projet car l'ajout d'une extension doit se faire sans complication et donc sans changer notre architecture. Nous considérons que l'ajout de nouvelles extensions correspondra à l'ajout de nouvelles tuiles, règles, façon de compter les points ou de nouveaux meeples.

I - Liste des tâches

Au cours des différents échanges, nous avons listé un certain nombre de tâches concentrées autour de l'architecture de notre projet. En effet, nous considérons la mise en place de l'architecture critique puisqu'une bonne architecture facilitera grandement l'implémentation. Elle doit donc être réalisée en priorité. Nous avons identifié différents aspects à approfondir par rapport au travail déjà effectué : (voir III, UML).

1 - Réfléchir de manière concrète à l'architecture par rapport aux différents types de tuiles/meeples (leurs types, les paysages contenus dessus etc..) et des extensions

2 - Étudier les différents designs patterns, leur pertinence et intérêt selon le contexte et en conséquence leur potentielle implémentation dans notre architecture

3 - Réfléchir à l'ajout de fonctionnalités (les différentes extensions) et aux modifications nécessaires à notre architecture en conséquence (exemple : différents constructeurs pour Pioche selon extension choisie ?)

4 - Commencer l'implémentation de ce dont on est certains (éléments principaux représentant la structure du jeu)

5 - Apprendre le bon fonctionnement de Qt, une API orientée objet et développée en C++, (Composants d'interface graphique (widget)) et faire une première maquette de notre interface

II - Répartition des tâches restantes

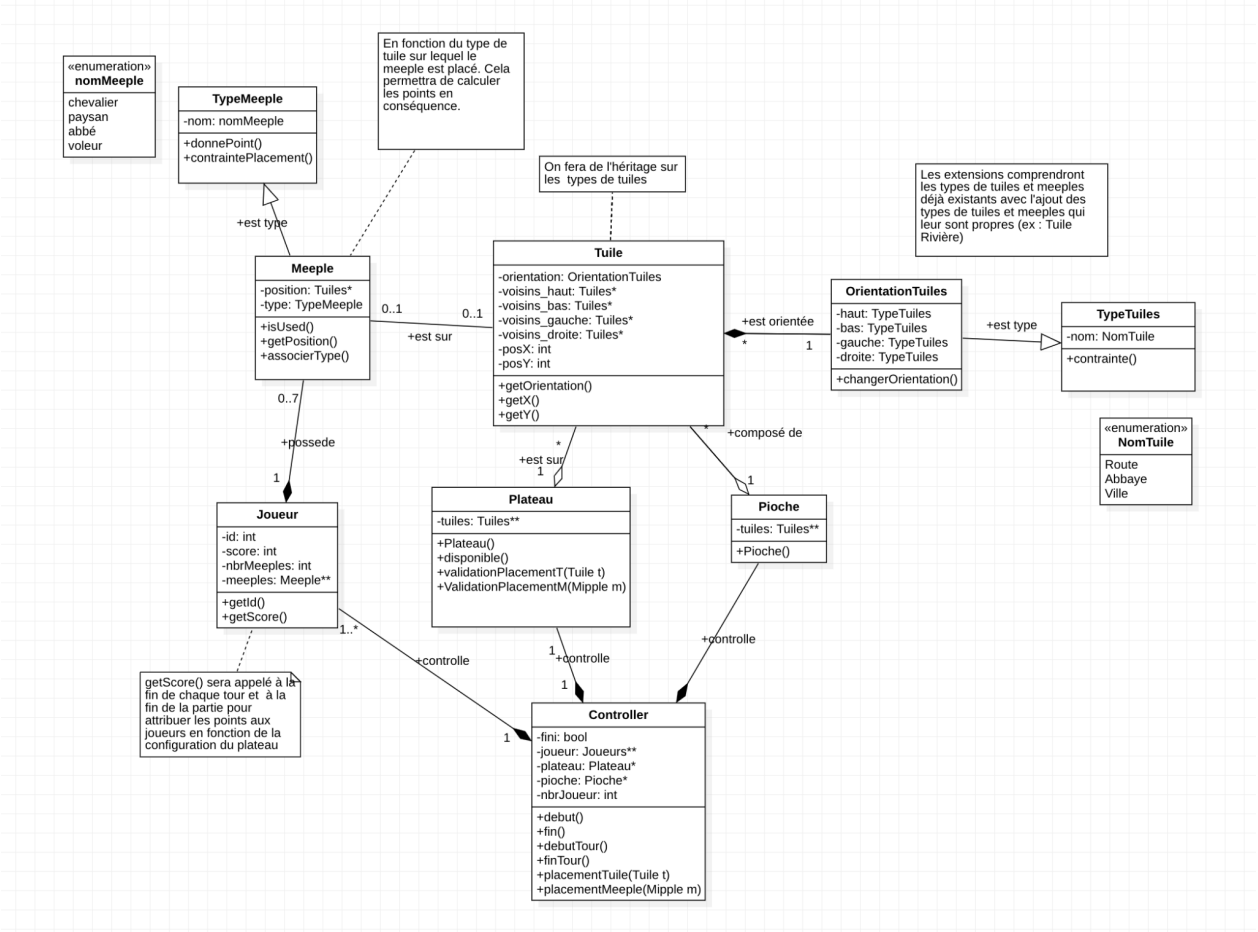
La réflexion sur l'architecture sera faite pour tout le monde afin d'échanger sur nos différentes compréhensions et approches des problématiques (design pattern, ajout des extensions). D'une importance critique, cette réflexion doit être faite par l'ensemble du groupe dans le but de maximiser la pertinence de notre proposition.

L'implémentation n'étant pas pour le moment d'une importance majeure, sa répartition sera décidée au moment opportun. Une fois l'architecture validée, nous définirons plus spécifiquement celle-ci.
On commencera tout de même à implémenter au fur et à mesure les éléments de structure du jeu (Tuiles, Pioche, Plateau contrôleur etc..) représenté dans notre UML une fois validés afin de gagner en efficacité.
Kelyan s'occupant de la partie Qt, il s'occupera de l'approfondissement de Qt ainsi que de la proposition de maquette de l'interface graphique.

III - Répartition et durée des tâches déjà effectuées.

La réussite d'un projet doit partir sur de bonnes bases. Plusieurs heures de réunion ont été alors mises en place. Cela a commencé par l'analyse des règles du jeu à travers des écrits et vidéos, puis avec des parties entre nous et seul. Cela nous a pris environ 4h en sachant qu'une partie est comprise entre 30min et 1h. Après une bonne compréhension du jeu dans sa globalité, il fallait se mettre d'accord sur les différentes conventions de développement. Grâce à ces conventions, Massil et Ambroise ont pu réaliser un premier diagramme UML qui leur a pris environ 4h. Plusieurs réunions ont été nécessaires pour mettre en commun les nouvelles idées sur le diagramme et corriger les erreurs si nécessaire. Enfin, le présent document à été produit par l'ensemble du groupe.

1 - UML



2 - Note de clarification UML

Le jeu Carcassonne est basé sur le placement de tuiles. Il nous a donc semblé évident de créer une classe Tuile. Cette classe est composée de 3 types d'attributs : 2 attributs pour expliciter la place qu'aura la tuile sur le plateau, 4 attributs représentant les tuiles voisines. Si une tuile n'a pas de voisin d'un côté alors nullptr sera affecté à l'attribut en question. Enfin, 1 attribut permet de connaître l'orientation de la tuile. Nous considérons qu'une tuile est caractérisée par 4 possibilité de types différents, un type sera attribué à chaque côté de la tuile. Cela sera possible grâce à la classe OrientationTuile qui possède 4 attributs, 1 pour chaque côté. Les différents types de tuiles seront gérés à l'aide de la classe TypeTuiles qui sera la base pour ajouter des fonctionnalités liées au différents types de tuile.

Les tuiles seront placées sur un plateau, représenté dans notre architecture par une classe composée d'un tableau de pointeur vers des Tuiles. La classe Plateau sera chargée de la validation de placement d'une tuile et d'un Mipple.

En effet, les joueurs, représentés par la classe Joueur seront identifiés par un id, par leur score, par leur nombre de meeple à disposition ainsi qu'un tableau de

pointeurs vers des objets de la classe Meeple. La classe Meeple permet de définir un meeple, positionné sur une tuile ou non. Chaque meeple est associé à un joueur et à un type.

La classe TypeMeeple, de la même manière que la classe TypeTuiles permet de définir l'ensemble des fonctionnalités liées à un type de meeple (chevalier, paysan, abbé, voleur…). Les méthodes de ces 2 classes auront des comportements différents en fonction de leurs attributs nom, cela permettra de rajouter des tuiles et des meeples sans compromettre celles et ceux déjà existants.

Enfin la classe contrôleur permet de gérer une partie, la faire débuter, la finir, placer des tuiles et des meeples sur le plateau. Le contrôleur contrôlera donc la pioche, le plateau et les joueurs

3 - Convention de développement

Nous sommes conscients que de bonnes pratiques de développement contribue grandement au bon déroulement et à la maintenabilité d'un projet de génie logiciel. C'est pourquoi il nous a semblé essentiel de définir dès le début du projet, avant de commencer à coder, certaines conventions de développement.
Tout d'abord, l'ensemble des noms attribués à des méthodes, fonctions, classes, variables seront écrites en français.
Nous avons également défini des conventions de nommage de nos variables. Nous utiliserons la notation PascalCase pour nommer nos classes et énumérations. Nous utiliserons la notation camelCase pour nommer nos fonctions et variables.
À titre d'exemple : UneClasse{}, UneEnumeration, uneFontion(), uneVariable.
En ce qui concerne l'ensemble des fichiers que nous serons amené à créer leurs noms seront sous la forme nom_du_fichier.extension

4- Outils et logiciels utilisés

Les différents choix logiciels et applications utilisés jusqu'à présent sont :

- Git : Nous avons décidé de choisir Git qui est un logiciel de gestion de version. Git permet un travail en équipe en utilisant tous les mêmes fichiers et suit les modifications que nous apportons aux documents, de sorte que nous disposons d'un enregistrement de ce qui a été fait et que nous pouvons revenir à des variantes explicites si nous en avons besoin. Git simplifie également la coordination des efforts, en permettant que les modifications apportées par plusieurs personnes soient toutes converties en une seule source tout en évitant des confusions.

- GitHub : Nous avons décidé d'utiliser GitHub car cela nous permet de collaborer et de sauvegarder notre travail très facilement. De plus, plusieurs

personnes dans le groupe connaissaient déjà bien ou en avaient entendu parler, ce qui a facilité le choix.

- Google drive : Nous avons décidé d'utiliser Google Drive pour la préparation des livrables car elle nous offre le moyen le plus simple de stocker, synchroniser et partager nos fichiers en toute sécurité, en résolvant les problèmes fondamentaux de la collaboration en équipe. Donc cela est plus efficace et productif en termes de travail.

- StarUML : Nous avons décidé d'utiliser StarUML Logiciel car ce logiciel nous a été conseillé par Massil, et permet de créer des diagrammes UML en utilisant une interface graphique aisément.

- Qt : Nous avons décidé d'utiliser Qt qui est une framework utilisé pour concevoir des interfaces graphiques, et l'architecture d'une application en utilisant des mécanismes de signaux et slots. Ce logiciel nous a été introduit en TD, ce qui nous donne une bonne première vision de Qt.

- Visual Studio Code : Nous avons décidé d'utiliser Visual Studio Code car c'est un éditeur de texte installé sur les ordinateurs de l'UTC et de plus est utilisé majoritairement dans notre groupe. Elle permet la compilation "à la main" du C++.

- Discord : Nous avons utilisé Discord afin de créer un serveur spécifique consacré au projet permettant de discuter et/ou s'appeler et aussi de déposer nos idées dans les différents canaux mis en place.

IV - Cohésion de groupe et implication de chacun

Notre groupe s'est formé assez naturellement en fonction des rencontres de chacun lors du TD de LO21. Nous ne nous connaissions pas avant de commencer le projet. Nous avons eu l'occasion de nous découvrir lors de séances de jeu pour découvrir Carcassonne. Ces moments furent l'occasion d'échanger à propos de nos expériences respectives et de nos différentes compétences. Aucun d'entre nous n'avait codé en C++ avant de suivre LO21. Certaines personnes (Ambroise, Sylvain, Taoufiq) avaient déjà utilisé git auparavant, sans en avoir une maitrise avancée.

Nous avons très rapidement pris conscience que nous nous lancions dans un projet impressionnant et qu'il serait primordial de nous organiser de manière efficace pour ne pas prendre de retard et fournir un livrable finale de la meilleure qualité possible.

Étant donné le stade d'avancement du projet, il était difficile de se répartir les différentes tâches à réaliser. Nous avons réalisé une partie du travail ensemble, notamment la réflexion liée à notre architecture, la pratique du jeu. Certaines

personnes ont ensuite réalisé des tâches plus particulières tel que l'élaboration de l'UML en accord avec l'architecture discutée, la création d'un document listant l'ensemble des contraintes liées aux règles du jeu qu'il sera primordiale de prendre en compte.