Monopoly A La Ferme

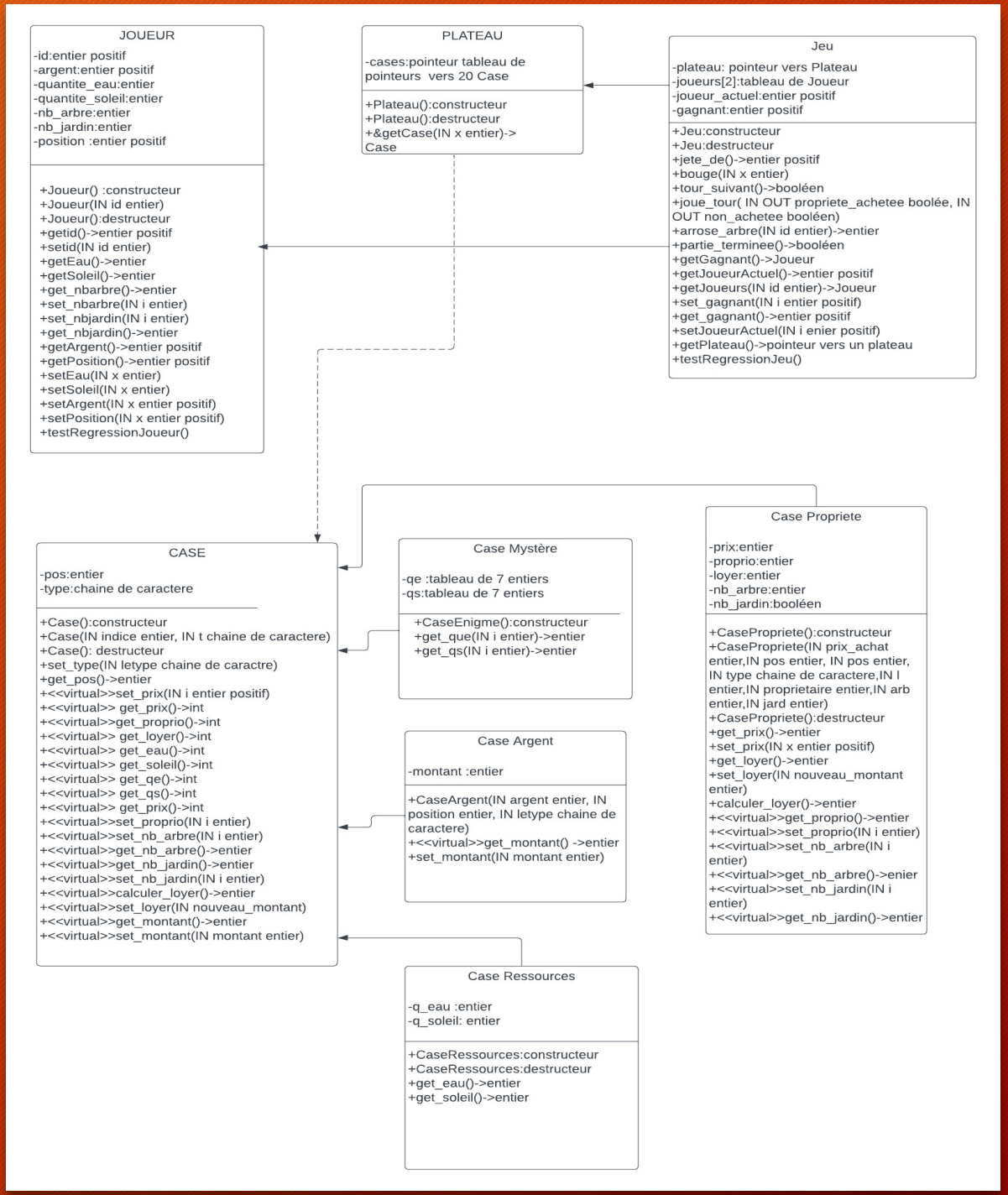
AZIZ Jane 12102430
KARAM Roudy 12018275

Principe du Jeu

Similaire au Monopoly traditionnel, mais à la place des maisons les 2 joueurs doivent acheter des terrains et les planter et aussi ils doivent les entretenir en les arrosant et en leur donnant du soleil. Il pourra gagner et perdre les ressources tout au long du jeu.

UML

- Le diagramme des classes modifié.



Quelques Classes

Jeu.cpp:
Dans la classe Jeu, on a un constructeur qui va initialiser 2 joueurs(0 et 1), et on met le gagnagt à -1 et c'est le joueur 0 qui va commencer la partie.
On a ecrit la fonction 'jete_de' qui va donner des valeurs aléatoires pour le de.
La fonction bouge va déplacer les joueurs en fonction de la valeur du dé que le joueur obtient apres avoir cliquer dessus.
La fonction 'tour_suivant', alterne entre le joueur 0 et 1 , pour que chacun puisse avoir son tour.
La fonction 'joue_tour' va à chaque fois modifier les données des utilisateurs en fonction du type de la case surlaquelle il est tombe apres avoir jeter le de.

Suite de la classe →

Quelques Classes

- S’il tombe sur Case Ressources le joueur actuel va perdre ou gagner 3 unités d’eau et de soleil.
- S’il tombe sur Case Argent le joueur actuel gagne ou perd 200 euros.
- S’il tombe sur la Case Mystere le joueur pourra perdre entre 0 et 3 unites de soleil et d’eau, ou gagner entre 0 et 3 unites d’eau et de soleil.Si le joueur actuel gagne son adversaire perd et vice vers ca.
- S’il tombe sur la Case Propriete, il faut vérifier plusieurs choses, s’il est le proprietaire, sinon s’il veut acheter la propriete en verfiant qu’il a l’argent suffisant pour l’acheter,s’il est déjà propriétaire il faut lui demander s’il veut acheter un arbre ou un jardin. S’il tombe sur une propriete de son adversaire, il doit payer les taxes de passage qui augmentent par rapport au nombre d’arbres ou bien terrain sur ce terrain.

Aussi, dans cette classe on pourra dire qui est le perdant ou le gagnant.

Quelques Classes

Plateau.cpp:

Dans cette classe on a initialisé le plateau formé de 20 cases. Pour cela on a créé un tableau de pointeurs vers Case.

Vu que Case a plusieurs héritages qui sont les différents types de Case dans notre jeu (CasePropriete,CaseArgent,CaseEnigme…) on devait relier chaque case qu'on a créée à son type.

On a d'abord initialisé toutes les cases sur le tas avec le type de ;a case qui convient à l'indice i. Et pour chaque case on a utilisé le 'dynamic_cast' pour pouvoir relier la case,à son type spécifique (CasePropriete, CaseArgent, CaseEnigme…) , afin de pouvoir utiliser les fonctions qu'on a créer pour chaque héritage et pour bien implémenter et manipuler les cases.

Quelques Classes

Case.cpp:

Dans cette classe on a le constructeur et destructeur ainsi que les fonctions dont on a besoin pour manipuler les cases qu'on a initialise dans la classe plateau.cpp.

Vue que les cases chacune est de type different(les heritages), avec le dynamic_cast dans plateau.cpp, on a besoin d'appeler les fonctions qu'on a créé dans chaque classe heritage et qu'on va utiliser pour chaque case, dans Case.cpp.

Conclusion

Nous avons passé beaucoup de temps sur ce projet en dehors des heures de TP. Mais on a pu exécuter ce qu'on voulait. Toutes les fonctionnalités qu'on souhaitaient implémenter marchent exactement comme ou veut.

On a eu du mal au début avec SDL2 et l'affichage texte des valeurs qui sont modifiées à chaque tour mais on a trouver la solution en cherchant sur internet. Aussi, on a eu un peu de mal pour séparer SDL2 des autres fonctions mais on a réussi à le faire. De plus on a eu quelques erreurs de segmentation mais on a pu, en travaillant ensemble, à comprendre les problèmes et les résoudre.

Si on disposait de plus de temps on aurait ajouter plus de fonctionnalités au jeu et on aurait mis la possibilité d'avoir plusieurs joueurs au lieu de deux uniquement.