Rapport

YE Nicolas 71814766
ROUGEOLLE Yoan 21953845

2022-2023

1 Introduction

Ce projet a été réalisé dans le cadre du module Langage à Objet Avancé de 2022-2023. L'objectif est d'implémenter les jeux suivants en C++ avec une interface graphique : Domino, Trax et Carcassonne.

2 Déroulement du projet

Cette partie regroupe ce qu'on a traité ainsi que les aspects les plus significatifs.

Tout d'abord on commencé par concevoir un début de modèle (pour Domino et Trax). Donc on a de suite implémenté les classes Piece, DomPiece, TraxPiece. On a dû un peu réfléchir pour la représentation du plateau. On a un peu hésité sur sa stucture, mais on choisi de faire un tableau dynamique à indice entier (positif et négatif). Et donc on a opté pour "étendre" la classe vector et faire une classe VecZ. Puis on a fais la classe Game, où on a fait les fonctions canPlace et place, qui sont les 2 fonctions principales du jeu qui sont étendues par la suite par les différents jeux. Notamment les calculs de connexité au moment de placer une pièce sont fait directement dans Game. Et donc pour le Domino, il presque fallut rien faire de plus, le jeu fonctionnait déjà. Pour Trax ça a été un plus compliqué pour le parcours à faire, il fallait stocker plus de variables et puis les coups forcées qu'il fallait calculer.

Nous avons en parallèle commencé à construire notre interface graphique : Structure en différentes scènes, menu principal, plateau (Zoom, déplacement de la "caméra", placement de carré) ainsi que les animations (qui sont plus simple à faire comparé à d'autre librairies). Ensuite il a fallu afficher les pièces sur le plateau. Pour domino, il fallut tracer "à la main" les pieces. Et pour Trax, il a fallu tracer l'image, mais surtout gérer les rotations qui se font autour du coin en haut à gauche de l'image, et aussi de gérer la "synchronisation" entre la rotation de l'image et la rotation de la piece dans le modèle.

Puis il a fallu faire un écran de configurations de parties, ce qui a été plus compliqué que prévu de part le fait qu'il a fallu créer nos classes pour nos éléments graphiques. Nous avons implémenté une classe Button qui fait usage de fonction passées sous forme de lambda pour exécuter des actions et une classe TextField qui gère le focus ainsi que les entrées clavier.

Ensuite, nous nous sommes penché sur le Carcassonne. Nous nous sommes concentré sur l'algorithme de parcours ainsi que sur la structure que devaient avoir les pièces pour pouvoir représenter le niveau de détail de celles-ci et qu'elle soit adaptée au parcours. L'interface graphique a été complétée après que nous ayons implémenté tout le modèle. Donc il a fallut tracer les pièces de Carcassonne. L'affichage se fait comme pour Trax, mais il imposait aussi de pouvoir

placer les pions sur la pièce, ce qui n'a pas été une mince affaire.

3 Difficultés rencontrées

Première grosse difficulté a été de savoir utilisé la librairie SFML. En effet c'était la première fois que nous l'utilisions, et de plus cette librairie nécessite que l'on créée nous même nos éléments graphiques (Scene, Button, TextField). Ceci nous a pris énormément de temps pour faire quelques chose de joli.

Ensuite les 2 premiers jeux ont été implémenté sans trop de difficulté, en revanche pour Carcassonne cela a été une autre paire de manche. Le nombre de règles assez grand, en plus de la complexité de ces dernières, notamment celles qui concernent les parcours des plaines en fin de partie. D'ailleurs on a dû plusieurs fois refaire la structures des pièces de Carcassonne, parce que nous n'avions pas remarqué les particularités de certaines pièces.

Finalement, nous avons essayé de respecter la séparation MVC mais l'implémentation du Controlleur fût un échec et a été retiré du rendu final.

4 Extensions possibles

Nous pourrions envisager des variantes à nos jeux par exemple un Trax avec plus de 2 joueurs, un plateau plus grand et des tuiles plus complexes.

5 UML

Les images suivantes pourront être trouvées dans les fichiers rendus sous le nom de : UML-M.png et UML-V.png oubien diagram.drawio (nécessite draw.io)

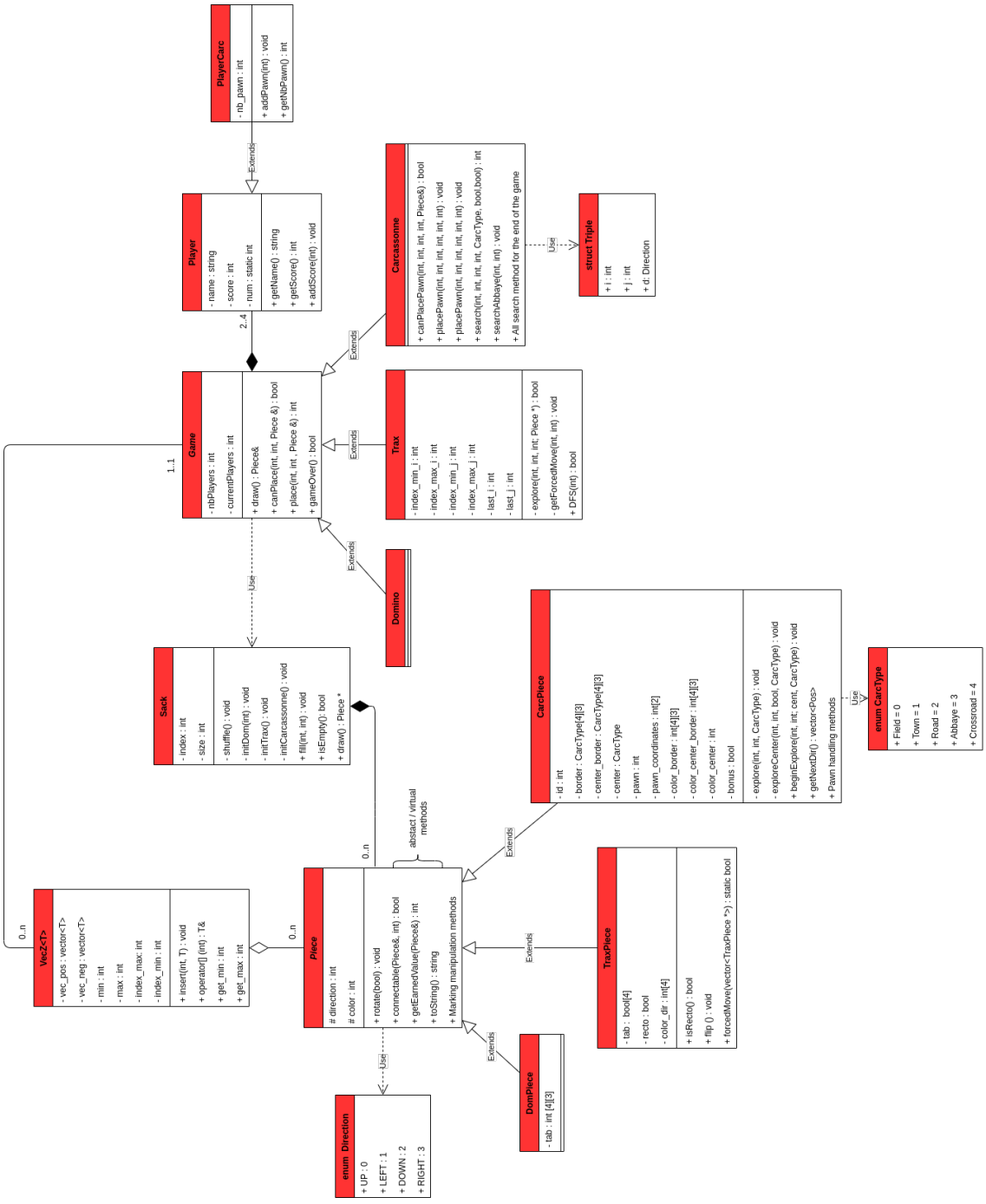

Figure 1: Modèle

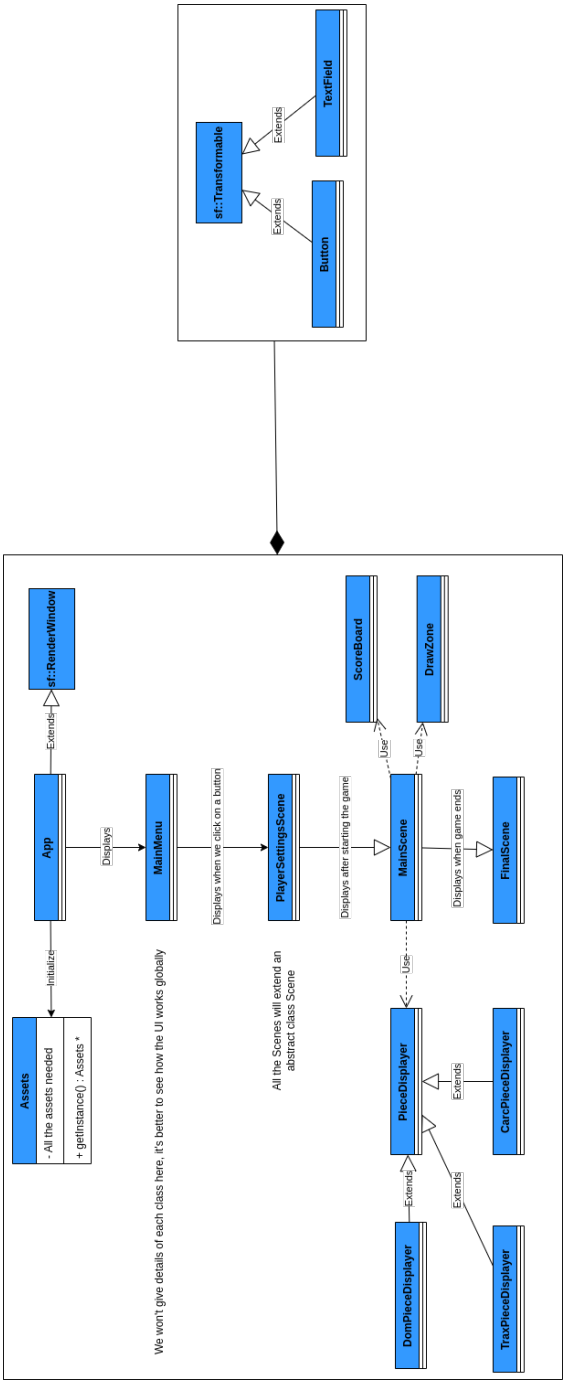
sf::Transformable
TextField
Button
Extends
Extends
sf::RenderWindow
Extends
App
Displays
MainMenu
Displays when we click on a button
PlayerSettingsScene
Displays after starting the game
MainScene
Displays when game ends
FinalScene
ScoreBoard
DrawZone
Use
Use
Initialize
Assets
- All the assets needed
+ getInstance() : Assets *
We won't give details of each class here, it's better to see how the UI works globally
All the Scenes will extend an abstract class Scene
Use
PieceDisplayer
Extends
CarcPieceDisplayer
DomPieceDisplayer
Extends
TraxPieceDisplayer
Extends

Figure 2: Vue