Athanasakis Alexandre | matricule : 42991
Laurent Kesier | matricule : 52167

Rapport Diaballik Remise 3

Application GUI

Professeur : Jonas Beleho (BEJ)

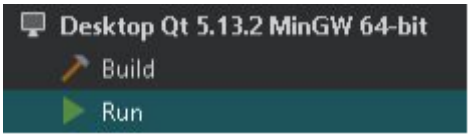
INTRODUCTION

Cette remise a comme objectif d'ajouter une application visuelle a nos classes métiers crées lors de la remise précédente. Nous avons, pour cela reçu un tutoriel vidéo afin de nous familiariser avec la création d'application sous qtCreator et nous nous sommes basés sur cela pour créer une application centrale qui permet l'interaction des widgets et du modèle.

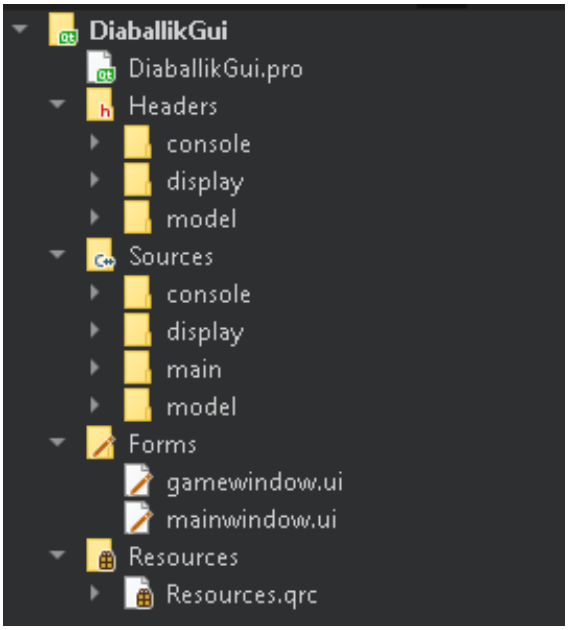
Dans ce rapport nous allons vous présenter le projet ainsi que les problèmes auxquels nous avons dû faire face lors de ce travail ainsi que la façon dont nous les avons résolus (si c'est le cas) ainsi que des bugs ou autres informations importantes à spécifier pour l'utilisation de l'application.

PRÉSENTATION

Version de Qt utilisée :



Structure du projet :



DiaballikGui.pro : Projet QWidget en c++17.

Console : contrôleur et vue de la remise précédente. Inchangés par manque de temps mais toujours fonctionnels.

Display : vue gui du projet regroupé en 3 classes 1 pour la page principale, 1 pour la page du jeu et la dernière représente les cases du jeu.

Model : classes métier du jeu avec pas mal de changements apportés pour rendre le code plus pertinent et plus homogène avec le gui.

Main : classe principale du code, elle contient 2 méthodes main une pour la version console et l'autre pour la version gui.

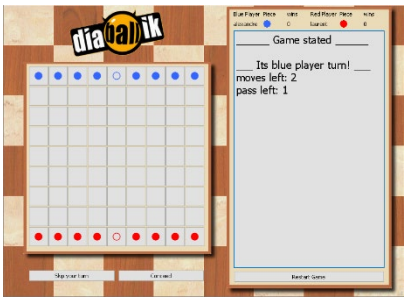
mainwinwow.ui : contient la page d'accueil du projet générée avec le gestionnaire d'applications de qtCreator.

gamewindow.ui : contient la page de jeu du projet générée avec le gestionnaire d'applications de qtCreator.

Ressources : contient les images utilisées par l'application, allant des pions au backgrounds.

FONCTIONNEMENT

Page d'accueil :
Pour commencer une partie, il faut tout d'abord indiquer le nom ou le pseudo des 2 joueurs, ensuite choisir le type de plateau désiré et enfin cocher ou non la variante avant d'appuyer sur le bouton « Launch Game ».



Page de jeu :

La partie se lance automatiquement, le joueur bleu commence, sélectionner une pièce du tableau afin d'effectuer une action ou cliquer sur les boutons au bas de la page. Vous pouvez aussi utiliser le menu du haut.

ÉCARTS, AJOUTS ET PRÉCISIONS

Notre projet ne possède pas les caractéristiques suivantes :

- Design pattern observateur observé : Nous avons décidé de nous passer du design pattern afin de générer un code basé uniquement sur le MVC (modèle vue contrôleur).

LES BUGS RESTANTS

Notre projet possède encore les bugs suivants :

- Antijeu partiellement fonctionnel : Notre code vérifie uniquement si 3 pions sont bloqués mais pas s'il n'est plus possible de passer.

JUSTIFICATIONS DES WARNING

- Pour la conversion de int a unsigned sur certaines versions de Qt il y a un avertissement sur d'autres non, nous avons décidé de ne pas corriger cela puisque chez nous ça ne l'affiche pas.

PROBLÈMES RENCONTRÉS

- Pour gérer la sélection de pions nous avons du ajouter les méthodes **cleanValidMoves**(), **cleanvalidPass()** et **cleanselected()** afin de permettre au joueur de sélectionner une pièces après l'autre, ces 3 méthodes servent a vider les listes et les position générée en tant qu'attribut dans le game pour la pièce courante lorsqu'elle change.

- Parfois les modifications apportée au fichiers gamewindow.ui et mainwindow.ui ne s'affichais pas en lançant le programme, il nous a fallu supprimer quelques fichiers .o ainsi que le fichier .moc afin de remédier à ce problème.

- Lors du lancement du code sur un autre pc, il est nécessaire de refaire un qmake et une compilation propre afin de pouvoir lancer le programme.

ESTIMATION DU TEMPS NÉCESSAIRE

Environ 80 heures à deux.