

Interactions Homme-Machine TP1

Prise en main

Si Qt Creator (IDE) n'est pas installé sur votre machine, commencez par installer puis lancer Qt Creator depuis <https://www.qt.io/offline-installers>. Installez également la dernière version de Qt5 (lib).

Alternativement, installez facilement une version moins récente via aptitude (qtcreator, qt5-default).

Qt Creator devrait être capable de trouver Qt5 automatiquement. Si ce n'est pas le cas, allez dans le kit Desktop et changez le lien vers la version de Qt dans les paramètres.

Une fois Qt Creator installé, clonez le projet <https://git.unistra.fr/nicolas.lutz/tp-ihm-2019> et ouvrez *memory.pro* avec Qt Creator. Ouvrez le fichier *mainwindow.ui* : vous travaillerez avec cette interface très simple faite pour un seul joueur de memory.

Dans ce TP, nous séparons fortement le noyau logiciel du noyau interface graphique. Le noyau interface graphique doit s'appuyer sur le noyau logiciel, mais le noyau logiciel ne peut en aucun cas utiliser des éléments d'interface. Ce principe est appelé la "séparation des préoccupations" et permet de modifier l'interface sans changer le noyau logiciel, et donc de diminuer les contraintes - augmenter les degrés de liberté du code source.

Implémentation du noyau logiciel (jeu)

Commencez par implémenter le jeu du memory dans *memory.cpp* et *memory.h* :

1. Complétez la classe Card, qui représente une carte, en créant des membres public et/ou privés si nécessaire. Une carte a un id donné dans son constructeur qui représente son motif, mais il faut aussi déterminer si elle est face cachée ou face découverte, et éventuellement si sa paire a été trouvée.
2. Complétez la classe Memory, qui représente l'état du jeu lui-même :
 - Complétez *score()* en renvoyant simplement le membre *m_score*.
 - Complétez *NbCards()* et *getCard()* en utilisant les fonctions du *vector* *m_cards*
 - Complétez *setNbCards()* en effaçant et en ajustant la taille de *m_cards*, ainsi qu'en réinitialisant les autres membres. Pensez également à initialiser l'id du motif des cartes de façon séquentielle (2 cartes par id).
 - Complétez *revealCard()* qui révèle la carte à la position *cardNb*. Vous pouvez stocker un pointeur vers cette carte dans le membre *m_revealedCards* pour la retrouver facilement. Renvoyez *true* si le joueur a retourné une seconde carte donnant une paire, *false* dans tous les autres cas. Enlevez également 1 point au score si une paire n'a pas été trouvée après avoir retourné deux cartes, et ajoutez 10 points si une paire a été trouvée.
 - Complétez *hideRevealedCards()*, qui cache les cartes révélées si une paire n'a pas été trouvée.
 - Complétez *checkWinCondition()*, qui parcourt le vecteur pour vérifier si le jeu est fini ou non.
 - Complétez *shuffle()*, qui mélange les cartes (utilisez *random_shuffle()* avec l'*iterator* de *vector*).
3. Complétez votre code avec des *assert*, qui vous permettront d'identifier des erreurs d'appels (par exemple, appeler *getCard()* avec un id trop grand, ou utiliser *revealCard()* alors que deux cartes ont déjà été révélées).
4. Assurez-vous au moins que le code compile. Vous pourrez le tester avec le noyau interface.

Implémentation du noyau interface

Implémentez le noyau interface graphique correspondant à l'interface dessinée. Un élément d'interface graphique X créé dans Qt Designer est récupéré dans la classe correspondante à l'interface créée via *ui->X*. Vous devrez vous servir intensivement de la documentation de Qt. En général, il suffit de taper le nom de la classe que vous manipulez sur un navigateur et vous trouvez rapidement la page associée.

N'oubliez pas de tester votre interface au fur et à mesure.

1. Dans *on_pushButton_play_pressed()*, qui est déjà connecté au signal *pressed()* du bouton "Play" via le fichier *mainwindow.ui* :
 - Faites disparaître le bouton "Play" avec *setHidden()*.
 - Activez tous les boutons contenus dans *m_cardButtons* (ce sont des *QPushButton*) avec *setEnabled()*.
2. Dans *on_pushButton_reset_pressed()*, qui est déjà connecté au bouton "Reset" :
 - Faites apparaître le bouton "Play".
 - Désactivez tous les boutons de *m_cardButtons*.
 - Utilisez *value()* pour obtenir le nombre de cartes stocké dans la spin box, et :
 - Utilisez cette valeur pour initialiser le jeu avec *setNbCards*.
 - Allouez autant de *CardButton* dans *m_cardButtons* qu'il y a de cartes dans le jeu en utilisant *new* et *push_back*.
Remarque : *CardButton* est une classe fille de *QPushButton* qui a en plus un membre donnant sa position dans le jeu.
 - Connectez le signal *pressed()* de chaque bouton au slot *on_cardButton_pressed()*.
 - Enfin, ajoutez chaque bouton dans le layout "grille" réservé aux cartes avec *_unsignedToQSize* donnant la ligne (*.width*) et la colonne (*.height*) prévues pour la carte dans le layout en fonction de son id.
3. Dans *on_cardButton_pressed()* :
 - Révélez la carte sélectionnée s'il est possible de révéler la carte d'après la spécification du jeu dans l'état où celui-ci est. Dans un premier temps, vous pouvez afficher l'id du motif de la carte lorsqu'elle est révélée.
 - Vérifiez si le jeu est gagné ou pas après une paire trouvée.
 - Si une paire n'a pas été trouvée après deux cartes révélées, lancez le timer, qui va appeler périodiquement la méthode *on_timerHideCards_timeout()* (voir liste suivante).
4. Dans *on_timerHideCards_timeout()* :
 - Stoppez le timer.
 - Cachez les cartes révélées.
 - Assurez-vous également de gérer correctement les problèmes potentiels liés au timer dans les autres fonctions.
5. Si vous avez réussi à aller jusque-là, vous pouvez essayer de rajouter des images pour les cartes, de rajouter un indicateur de temps écoulé, un bouton d'aide...