Compte-Rendu :

Mode Exercice :

Fichier: eval.cpp, eval.h

Présentation :

Le mode exercice est un mode proposant à l'utilisateur de s'évaluer.

Celui-ci est accessible à partir du menu: mode → exercise.

Une fois le mode exercice activé, chaque action entreprise par le pilote sera analysée. Si ces dernières ne sont pas appropriées à la situation alors le pilote en sera pénalisé sur sa note finale.

Au terme de l'exercice le pilote se verra attribuer une note sur 10.

Le programme retourne alors en mode simulation, le mode défini comme « par défaut ».

Implémentation :

Le mode exercice fonctionne avec la technologie des signaux et des slots proposée par la bibliothèque « Qt ».

Il est implémenté en tant que classe « Evaluation » héritant de Qwidget.

Il dispose de 5 attributs privés :

-int mark, la note attribuée à l'utilisateur

-int mistake_nb, le nombre d'erreurs commises par l'utilisateur

-SystemeCarburant* sc, un pointeur sur le système carburant, la classe recueillant les informations de l'état du programme.

-QMap<QString, GenericTpev*>& scMap, la map du système carburant permettant d'acceder à l'état de chaque pièce du système.

-Log *log, un pointeur sur l'historique du système permettant ainsi d'afficher les informations de l'évaluateur à l'utilisateur.

Il dispose également de 4 méthodes publiques :

-Evaluation(SystemeCarburant *systemeC, Log *log), le constructeur

-~Evaluation(){}, le destructeur

- int getMark() ,int getMistakeNumber(), les 2 accesseurs des attributs dont ils portent le nom.

Afin de communiquer avec le reste du programme, la classe Evaluation utilise les « slots », elle en dispose de 9.

Un pour chaque bouton avec lesquelles le pilote peut interagir, autrement dit, un pour chaque choix du pilote. Ces slots sont en réalité des algorithmes représentant les méthodes principales de l'évaluateur.
Enfin un neuvième slot, « resolved() » connecté à un signal envoyé par les moteurs est chargé d'indiquer à l'évaluateur que l'exercice est terminé.

Le fonctionnement de l'évaluateur est le suivant :

A chaque clic de l'utilisateur sur un bouton afin de changer l'état du système, un signal est envoyé à l'évaluateur depuis l'objet dont l'utilisateur souhaite altérer l'état. Il est important de noté que ce signal est reçu avant que l'état soit altéré.

L'algorithme correspondant est alors appelé, évaluant l'état actuel du système et l'intérêt de l'altération voulue par l'utilisateur. Si cette dernière est correcte alors l'évaluateur affiche « Correct » sinon il informe l'utilisateur que son choix est incorrect et soustrait 1 point à la note finale.

Chaque fois qu'un nouveau moteur est alimenté, un signal est émis afin d'en avertir l'évaluateur. Si il s'avère que ce moteur était le dernier a alimenter alors l'évaluateur termine l'évaluation et en informe l'utilisateur en lui donnant sa note. Le mode exercice est alors quitté pour revenir au mode simulation.

Connexion :

Fichier : connexion.cpp, connexion.h

Présentation :

La connexion propose à l'utilisateur de créer un compte ou bien de se connecter à travers l'interface graphique.

Une fois connecté l'utilisateur à accès à la fonctionnalité « File » lui permettant de sauvegarder et charger des fichiers dans l'application.

La connexion dispose d'un onglet dans le Menu. On peut choisir de créer un compte, se connecter ou bien se déconnecter.

Implémentation :

La connexion est implémentée en tant que classe héritant publiquement de QWidget.

Elle dispose de 3 attributs privés :

-QFile *usersFile, le fichier contenant les noms d'utilisateurs ainsi que leur mot de passe

-bool isconnected, un booléen indiquant si un utilisateur est connecté ou non

-QString userName le nom de l'utilisateur connecté, s'il y en a un. Il est initialisé à NULL.

La classe dispose de deux méthodes permettant de créer l'interface graphique de connexion ainsi que de création de compte.

-void ConnectionInterface()

-void newAccountInterface()

Elle dispose également d'une méthode permettant de vérifier si l'utilisateur entré est présent dans le fichier.

-bool alreadyExists(QString name, QString password)

La classe connexion fonctionne également avec la technologie des signaux et des slots proposée par la bibliothèque « Qt » ainsi elle dispose de 3 slots et de 2 signaux.

Ces 3 slots sont en fait les méthodes de la classe permettant de créer un utilisateur, connecté un utilisateur et déconnecté un utilisateur. Ils sont connectés à des signaux envoyés lorsque leur bouton correspondant est pressé.

Les signaux quant à eux, permettent d'informer le programme lorsqu'un utilisateur se connecte ou se déconnecte.

-void isConnected()

-void isDisconnectedSignal()

Le fonctionnement de la connexion est le suivant :

Dans le menu, l'utilisateur a accès à l'onglet connexion lui proposant d'interagir avec 3 boutons.

1. S'il s'agit de la création de compte (« Register »), la méthode correspondante est appelée et propose son interface à l'utilisateur. Une fois le bouton de confirmation pressé, le slot correspondant est appelé. Celui ci vérifie que l'utilisateur n'est pas déjà présent dans le fichier utilisateur puis le cas échéant ouvre le fichier d'utilisateur en mode concaténation afin d'y entrer le nom et mot de passe donné par le nouvel utilisateur.
2. S'il s'agit de la connexion (« Connection »), la méthode correspondante est appelée et propose son interface à l'utilisateur.Une fois le bouton de confirmation pressé, le slot correspondant est appelé. Celui-ci ouvre cette fois le fichier d'utilisateur en lecture seule puis cherche ses données dans ce dernier. S'il le trouve, il le connecte à l'application avec un message de bienvenue, sinon il l'informe de l'échec de la procédure.
3. S'il s'agit de la déconnexion, le slot correspondant est directement appelé et annule la connexion actuelle en émettant un signal qui sera perçu par le programme.