PROJET C++ : SIMULATEUR AVION

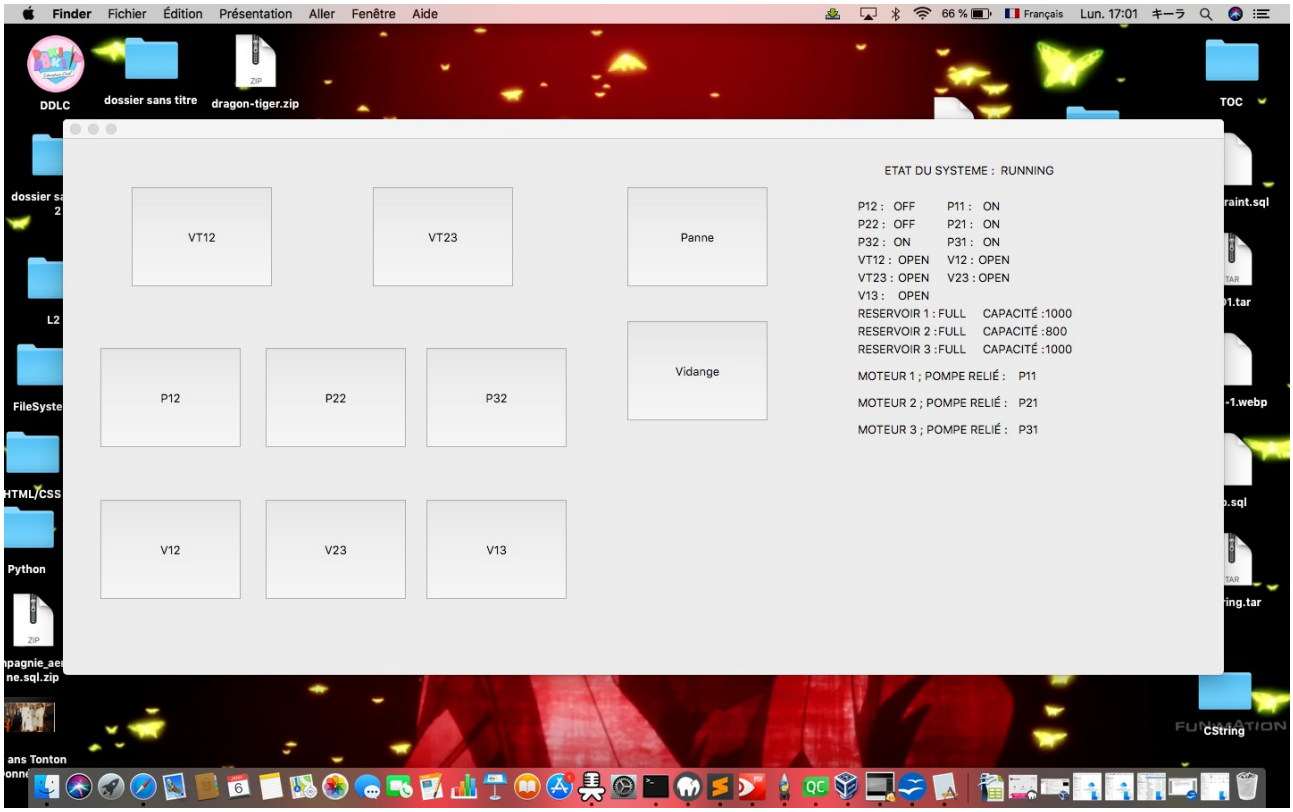
HU XUNYUE/ANDRIANINA RAKOTOMANGA

1. INTRODUCTION

Pour ce Projet, nous avons à implémenter une applications qui simule un vol d'avion, plus particulièrement le fonctionnement su système d'acheminement du carburant de manière simplifié.

Nous utiliserons donc le langage C++, Ainsi que le framework graphique Qt, qui permet de manière simplifié de créer une interface graphiques simple.

Notre projet devra inclure une programmation orientée objet, ainsi que des cas de tests pour les utilisateurs.



2.ARCHITECTURES

L'architecture Principale sera faite de Quatre Classe : Reservoir,Moteur,Pompe et Vanne. Les classes test et Fenêtres, seront utilisé pour l'implémentation de l'interface graphique.

Nous utiliserons des énumérations pour représenter l'état des Objet Crée

Pour les Pompes : enum etatPompe → ON/Ouvert, OFF/Fermer, BROKE/Casser

Pour les Vannes : enum etatVanne → OPEN/ouvert, CLOSE/fermer

Pour les Reservoir : enum etatReservoir → FULL/Rempli, EMPTY/vide

Reservoir.cpp/.h :

```
enum etatReservoir etat; Etat du Reservoir
int capacite; Capacité du Reservoir
int numero; Numero Du reservoir
int capmax; Capacité du reservoir
Pompe pompe1; Pompe Principale du reservoir

Pompe pompe2; Pompe de secours du reservoir
```

Vider() ; Vide le reservoir

Remplir() ; Remplis le reservoir

usePompe() ; Active la pompe eteinte et Eteinte la pompe Active

Pompe.cpp/.h

```
enum etatPompe state; Etat de la pompe

bool secours; Type de pompe vrai/secours faux/principale
 bool occupe; Disponibilité de la pompe

std::string nom; nom de la pompe
Reservoir* res; Reservoir relié à la pompe
```

Demarrer() ; Met la pompe sur ON

Eteindre() ; Met la pompe sur OFF

Casser() ; Met la pompe sur BROKE

isOccupee() Renvoie vrai si la pompe est déjà connecté à un moteur

Disponible() ; Met la disponibilité de la pompe sur vrai

Moteur.cpp/.h

```
int numeroMoteur; numero du moteur
Reservoir *ResAlimenter; reservoir qui alimente le moteur
Pompe* pompe; pompe qui alimente le moteur
```

Moteur(int numeroMoteur, Reservoir &res, Pompe &p) ;
numeroMoteur : Numero du moteur
res : Le reservoir connecté au moteur
p : la pompe connecté au moteur

Vanne.cpp/.h

```
enum etatVanne etat; etat de la vanne
std::string nom; nom de la vanne
```

3.IMPLÉMENTATIONS

L'objet Fenetre.cpp permet de gerer l'interface graphique ainsi que l'interaction entre les autres objet grâce au fonction qui permettent de faire communiqué les objet entre eux.

Ces objets sont stockés dans un Objet Test.cpp qui stockera donc Tank1,Tank2,Tank3, M1,M2,M3,les pompes et vannes associées

Les fonctions/slots ouvrir[nom de la pompe /pompe] permettent d'ouvrir ou fermer les vannes ou les pompes associer à leur noms.

Les fonctions check[nom de Vanne/Moteur/Reservoir] permettent de vérifier la validité du système, checkVT() permet également de rééquilibrer le niveau de capacité courante des réservoirs lorsque l'on ouvre une vanne inter_reservoir VT.

Les fonctions/slots panne/vidange() permettent de simuler une situations où une panne ou une vidange aléatoire se lance.

Les fonctions getUsers/subUsers() permettent d'inscrire dans la BDD le nom du pilote qui sera évalué.

Ces fonctions seront reliés à des boutons, initialisés dans le constructeurs de Fenetre.cpp. Nous avons utilisé la fonctions Qobject::Connect() pour pouvoir connecter les boutons au fonctions(slots).

4.APPLICATIONS GRAPHIQUES

Pour concevoir l'interface graphiques nous avons utiliser la bibliothèque Qt, notamment : QpushButton,QWidget,QLabel.

Nous avons utiliser des positions absolues pour l'emplacement des entités graphiques.

L'interface est simple mais claire, à gauche nous avons la partie d'interaction avec le pilote ou les boutons sont présenté selon le nom de l'objet auquelle ils sont relié et permet de soit ouvrir ou fermer ces objets.

La partie de droite correspond à l'affiche de l'état global du système. Il est mis à jour en temps réel.

Au milieu se trouve les boutons de simulations de panne et de vidange, Il suffit de cliquer sur l'un d'eux pour pouvoir actionner de manière aléatoire soit une panne , soit une vidange sur au moins 1 objet

Enfin le bouton terminer permet de quitter l'application et d'appliquer l'evaluation au pilote correspondant ;

5.CONCLUSION

Ce Projet fût une bonne expérience dans la programmation objet en C++. Cela nous as permis de comprendre et maitrisé le sujet en profondeur.

Cependant Nous n'avons pas pu complètement finir le projet. L'implémentation graphique étant opérationnel, mais la partie 2 sur la notation nous as semblé flou, nous avons donc simplifié l'implémentation en mettant une note de réussite ou de d'échec du test.