Rapport de Projet IHM

Membres du groupe : Jiangnan HUANG, Xiaoxuan HEI

Partie applicative:

Fonctionnalités:

1.Parser les fichier "airports.csv"

D'abord, on crée la classe "Airport" pour stocker les informations d'aéroports (nom, nomVille, nomPays, ICAO, latitude et longitude). Puis, On écrit la fonction "getAirport()" dans la classe "Systeme". Pour récupérer les informations, on utilise "FileReader("airports.csv")"

2. Récupérer la liste des villes d'un pays

On doit créer une liste de pays puis parcourt cette liste. Pour obtenir cette liste, on parcourt la liste d'aéroport et stocke les " nomPays". Puis, on crée une "HashMap<String,ArrayList<String>> villesD1Pays ". Les keys de cette HashMap sont les pays, et les values sont les ArrayList de villes de ce pays. La même démarche pour récupérer la liste des aéroports d'une ville.

3. Récupérer les avions en provenance ou à destination d'un aéroport

On crée la classe "ParserFlight" pour envoyer les requêtes. Dans cette classe, on écrit deux méthodes, la première "getFlight(String s)" est pour récupérer les avions en provenance ou à destination de l'aéroport dont le code ICAO est s. Puis on écrit deux fonctions "getFlightFromAir(String from)" et "getFlightToAir(String to)" (from et to sont les noms d'aéroports) dans la classe "Systeme" pour choisir les avions. La même démarche pour récupérer les avions en provenance et à destination de 2 aéroports. On met la fonction "getFlightFromToAir(String from, String to)".

4.Afficher dans la console les informations de vols

On parcourt les listes de vols et retourne les informations par les fonctions mentionnées précédemment.

5. Récupérer les avions en provenance ou à destination d'une ville

On utilise aussi la fonction "getFlight(String s)" ici. Puis on écrit deux fonctions "getFlightFromVille(String from)" et "getFlightToVille(String to)" (from et to sont les noms de villes) dans la classe "Systeme" pour choisir les avions. On doit changer les noms de villes aux codes ICAO d'aéroports d'abord.

6.Afficher dans la console des dernières positions d'un vol

Comme on a dit qu'il y avait deux fonctions dans la classe "ParserFlight". La deuxième "getPositionD1Vol()" envoie la requête avec le paramètre trFmt=f. On ecrit la fonction "getPosition(String s)". S est la ID du vol. On parcourt le resultat de la requête et trouve les informations sur ce vol. Enfin affiche les informations sur la

position.

7. Récupérer la liste des vols dans un rayon donné d'une position

On écrit la fonction "getFlightsIn(double longitude, double latitude)". Longitude et latitude sont les informations de la position donnée. Dans cette fonction on appelle aussi la fonction "getPositionD1Vol()" pour envoyer la requête. Quand la distance entre un vol de la liste et la position donnée est plus petite que le rayon donné. Après que l'on parcourt tous les vols dans la liste, on trouvera la liste des vols dont on a besoin.

Interface graphique:

Fonctionnalités:

1. Sélectionner un pays, une ville et un aéroport

Dans la partie applicative, on a déjà la liste des villes d'un pays et la liste des aéroports d'une ville. D'abord, on met la liste de pays dans un Combobox. Quand on choisit un pays, on récupère la liste de villes de ce pays en utilisant "HashMap villesD1Pays" et met la liste dans le deuxième Combobox. Quand on choisit une ville, on récupère la liste d'aéroports de cette ville en utilisant "HashMap airD1Ville". La liste d'aéroport va être mise dans le troisième Combobox. Puis on peut choisir un aéroport. Et puis, avec le nom d'aéroport, on peut lancer une requête pour afficher les vols. C'est la même chose pour le départ et la destination des vols.

Si on change un pays, le Combobox de villes et le Combobox d'aéroports vont être vidés. Si on change une ville, le Combobox d'aéroport va être vidé.

2. Afficher des sphères à l'emplacement de chaque aéroport

Quand on choisit un aéroport comme départ, une sphère jaune va être affiché sur la terre. Ici on utilise la fonction "displayTownDepart". C'est la même démarche pour la fonction "displayTownArriver" pour afficher la sphère rouge d'arrivée.

3.Afficher les listes du vol

Comme on a dit dans la question 1, quand on choisit un aéroport, il va lancer une requête et puis retourner une liste de vols, on a choisi l'identifiant et le type du vol pour signifier un flight. Toutes les listes seront affichées dans les deux ListViews myDepart et myArriver, un pour les vol de départ et l'autre pour les vols d'arrivé.

4.Afficher les vols dans la vue 3D

On a crée deux Groups de javafx pour stocker les modèles qui affichent les vols dans la vue 3D sur la terre. Comme on n'a pas réussi à utiliser le plane.obj, notre programme va créer des sphères quand il parcourt la liste des vols et stocker la sphère dans son group, et après il va ajouter le groupe dans le root3D for afficher tous les vols dans la vue 3D.

En plus, chaque fois on appuie le bouton Check, le programme va effacer les vieilles sphères et afficher encore une fois les vols actuel.

5.Mettre un vol en évidence

Comme on a mis tous les vols actuels dans les deux ListViews , quand le utilisateur clicke sûr un vol dans un ListView, le programme va appeler les fonctions myDepart.setOnMouseClicked et myArriver.setOnMouseClicked, les fonctions vont distinguer le vol qu'on a choisi et en plus recréer une sphère plus grand et en highlight pour lui, cette sphère va couvrir la vieille sphère, avec cette façon on peut mettre ce vol choisi en évidence dans la vue 3D.

5.Afficher les infos concernant l'avion sélectionné

On a créé un TextArea dans javafx pour afficher les infos du vol sélectionné, ça fonction au même temps qu'on appelle les deux fonctions myDepart.setOnMouseClicked et myArriver.setOnMouseClicked. Le programme va distinguer le vol qu'on a choisi et en plus parcourir ces infos (dans la classe Flight) et les stocker dans un argument String, et finalement affiche le String dans le TextArea qu'on a créé.

6.La trajectoire

On a trouvé une façon pour afficher la trajectoire, mais il reste quand même des bugs. On va expliquer cette partie dans la soutenance.

Il reste encore des bugs qu'on n'a pas encore regelé :

1. On n'a pas réussi à utiliser le plane.obj pour signifier les vols à cause de la rotation du modèle. Et en plus le plane.obj est toujours un peu décalé de la position qu'on l'a mis.
2. On n'a pas réussi a effacer les itinéraires qui sont déjà existé sur la terre.