Rapport de Projet : Visualisation Interactive des Aéroports et des Vols

Introduction

Ce projet consiste en la création d'une application interactive permettant de visualiser une représentation 3D de la Terre, d'afficher les aéroports les plus proches en réponse à des clics utilisateur et de récupérer en temps réel des données de vols grâce à une API. Le projet a pour objectif principal d'exploiter des données géographiques et aéronautiques en utilisant notamment le traitement graphique avec JavaFX.

Objectifs du Projet

Visualisation de la Terre en 3D : La sphère terrestre est représentée en trois dimensions (X, Y et Z) à l'aide de JavaFX, et une texture détaillée est appliquée pour donner l'illusion d'une vue réelle.

Localisation des aéroports : En cliquant sur la surface de la Terre, l'utilisateur peut afficher des sphères rouges à l'emplacement des aéroports les plus proches.

Données de vols en temps réel : L'application interroge une API externe pour afficher aéroports de départ avec des sphères jaunes.

Structure du Code

Classe Earth

La classe Earth représente la Terre en 3D. Elle crée une sphère avec un rayon de 300 pixels, applique une texture issue d'une image de la Terre, et gère les rotations autour de l'axe Y. Elle fournit des méthodes pour positionner des sphères colorées représentant les aéroports :

- **`displayRedSphere`** : Affiche une sphère rouge pour un aéroport sélectionné.
- **`displayYellowSphere`** : Affiche une sphère jaune pour les aéroports de départ.

Chaque sphère est positionnée selon les coordonnées géographiques des aéroports, transformées en coordonnées adaptées au bon format à l'aide de calculs mathématiques adaptés.

2. Classe World

La classe World gère le chargement des données des aéroports à partir d'un fichier CSV. Les informations essentielles telles que le nom, les coordonnées géographiques, et le code IATA des aéroports sont stockées dans une liste. Cette classe inclut :

- **`findNearestAirport`** : Recherche l'aéroport le plus proche d'une paire de coordonnées géographiques.
- **`findByCode`** : Permet de retrouver un aéroport à partir de son code IATA.

3. Classe Flight

La classe Flight modélise les informations relatives aux vols, tel le code IATA de l'aéroport. Ces données sont utilisées pour identifier les aéroports associés aux vols récupérés depuis l'API.

4. Classe JsonFlightFiller

Cette classe est responsable de l'analyse des données JSON reçues depuis l'API. Elle extrait les informations des vols et les transforment en objets Flight. Elle utilise des bibliothèques de parsing JSON pour traiter efficacement les réponses de l'API.

5. Classe Interface

La classe principale Interface gère l'affichage de la scène 3D, la gestion des interactions utilisateur et la communication avec l'API. Parmi ses fonctionnalités :

- Réception des clics utilisateur pour convertir les coordonnées (latitude et longitude).
- Recherche et affichage des aéroports proches avec une sphère rouge.
- Récupération des données de vols en temps réel à partir de l'API **aviationstack**.
- Affichage des sphères jaunes pour les aéroports de départ des vols.

Résultats Obtenus

- Une interface fonctionnelle affichant une Terre en rotation.
- La possibilité d'interagir avec la Terre en cliquant pour localiser les aéroports les plus proches.
- L'intégration des données en temps réel des vols, avec des sphères jaunes indiquant les aéroports de départ.