Enzo Gallo

Description fonctionnelle et technique du travail réalisé :

Différence avec le diagramme UML initial de la partie applicative :

Lors de l'implémentation de la partie applicative, j'ai tenté de suivre mon diagramme UML de départ mais je me suis perdu avec mes différentes Hashmap. En conséquence, je les ai remplacées par des ArrayList sauf listeAnneeAno qui est une Hashmap, ayant comme clé l'année et comme valeur l'anomalie, intégrée dans la classe Coordonnées. C'est cette Hashmap qui récoltera toutes les anomalies du fichier csv. En modifiant tout cela, j'ai perdu l'utilité de ma classe Années donc je l'ai supprimé. J'ai également ajouté des méthodes comme yearNumber() pour valider les tests du SimpleTest.

Description du code :

- *Partie applicative*

Classe CsvFileReader :

- Pour l'idée générale de conception de readCSV(), j'ai réutilisé et réadapté la méthode que j'avais réalisé dans un précédent projet.
- J'ai initialisé manuellement listeAnnees contenue dans la classe Earth car je manipule seulement le fichier à partir de la 2^{e} ligne et donc je ne traite pas les années données en tête de colonne, soit la ligne ''1880'',''1881'', …, ''2020''.
- Le séparateur est ici seulement la « , ».
- Je stocke mes anomalies dans la Hashmap en changeant dès le départ le type des anomalies en float.
- Dans le même temps, on stocke les latitudes et les longitudes dans un objet Coordonnées qu'on place dans une ArrayList (listeZones contenue dans Earth).

Classe Coordonnées :

- On a la latitude, la longitude et la Hashmap.
- J'ai implémenté un constructeur pour que ce soit plus simple par la suite pour déclarer les objets Coordonnées.
- J'ai réalisé une méthode equals car ça aurait pu m'être utile mais je ne crois pas l'avoir utilisé dans mon projet.
- Puis j'ai instancié des setters et des getters.

Classe Earth :

- On a deux ArrayList contenant respectivement les années et les zones. Je les ai créées en static pour les utiliser de manière plus efficace dans le package.
- J'ai réalisé les méthodes demandées (valeurMinAno(), valeurMaxAno(), yearNumber(), getAno(lat,lon,année), getAnoAnnee(année), getAnoZones(lat, lon)).
- J'y ai placé ma fonction main avec le lancement de la méthode readCSV().

- *Partie graphique*

Classe Main :

- Lancement de l'application Java FX.

Classe CameraManager :

- Récupérée de mon tutoriel.

Classe Modele :

- J'ai créé cette classe afin de stocker mes booléens qui gèrent l'activation des différents boutons et donc dans le but d'être un appui pour la classe Controller

Classe Controller :

- J'ai initialisé tous les objets présents sur mon application et dans mon fichier FXML.
- Mes méthodes click… gèrent la « prise de parole » des boutons afin de ne pas activer deux boutons successivement.
- Les méthodes createLine, geoCoordTo3dCoord, displayTown et AddQuadrilateral ont été soit données soit réalisées dans le tutoriel.
- J'ai créé la méthode coord3DToGeoCoord qui a donc le but opposé à celui de geoCoordTo3dCoord.
- La méthode afficheQuadri permet d'afficher un quadrillage coloré en fonction des anomalies. On vérifie donc que l'année entrée en paramètre est bien dans notre intervalle puis on initialise les textures. Ensuite, pour toutes les zones, on récupère l'anomalie avec la méthode getAno ainsi que les latitudes et longitudes associées aux zones dans listeZones. En fonction de l'anomalie on affecte la bonne texture à chaque carré du quadrillage. J'ai ici copié collé à chaque fois la déclaration des 4 Point3D et l'appel à la fonction AddQuadrilateral afin d'être sûr de ne pas appliquer la mauvaise couleur au mauvais endroit.
- La méthode afficheHisto repose sur le même principe mais on déclare un cylindre avec createLine dont d'ailleurs la hauteur a été ajusté en fonction de la valeur de l'anomalie, c'est-à-dire que deux cylindres de même couleur peuvent avoir des hauteurs différentes car la valeur de leurs anomalies sont différentes mais sont comprises dans le même intervalle.
- Dans la méthode initialize, j'ai repris comme base le code du tutoriel. J'ai ensuite créé une animation pour faire défiler les anomalies en fonction des années. Je récupère la valeur de la vitesse entrée dans le textFieldAnnee que j'ai utilisé pour diviser le temps t et ainsi contrôler la vitesse. Le slider et le textField des années suivent l'animation.
- J'ai ensuite réalisé les eventHandler classés par type dans mon code :
    - Pour le bouton radio Quadrilatères, je relance l'affichage de la Terre à chaque fois pour ne pas superposer les affichages. Si j'appuie sur ce bouton, il lancera directement l'affichage pour l'année 1880 car c'est la valeur initiale du slider.
    - Même chose pour le bouton radio Histogrammes
    - Pour le bouton Start, je lance l'animation précédemment expliquée mais j'ai un souci. En effet, lorsque je lance l'application, le timer s'enclenche sans même que j'aie cliqué sur le bouton start et donc si j'appuie tard sur start, par exemple 20 secondes plus tard, le timer a déjà avancé et donc au lieu de voir comme première anomalie celle de 1880, je verrais celle de l'année 1900 (pour une vitesse à 1 année par seconde). En conséquence, si j'attends trop, je peux ne rien voir puisque je stoppe l'animation lorsqu'on dépasse 2020 et il faut relancer l'application.
    - Même problème avec le bouton Pause, lorsque je fais pause, je reste figé par exemple à l'année n, ce qui est mon but mais, lorsque je veux relancer

l'animation, le timer a défilé et je ne vois pas l'année n+1 mais l'année n+20 si j'ai attendu 20 secondes (pour une vitesse à 1 année par seconde).

- Le bouton Stop stoppe bien l'animation et relance l'affichage initial de la Terre.
- J'ai affiché lors d'un clic sur la Terre les coordonnées géographiques associées en utilisant un PickResult que j'ai trouvé dans la documentation et j'ai suivi la logique des latitudes et longitudes pour implémenter cet eventHandler.
- On peut afficher les anomalies correspondantes à une année entrée dans le textFieldAnnee. Si on écrit une valeur plus petite que 1880, on affichera la valeur minimale soit 1880 car le slider suit l'année entrée mais se bloque sur sa valeur minimale. Même chose si on dépasse 2020.
- Pour les textFieldLat et textFiledLong lorsqu'on écrit des coordonnées, cela affiche le lieu en question sur le globe (grâce à displayTown).
- Le code pour le textFieldVitesse permet uniquement de lancer l'animation si les autres paramètres sont déclarés.
- Pour le slider, on peut le faire glisser et cela lancera l'affichage des anomalies pour l'année souhaitée.
- J'ai tenté d'afficher le graphique mais je n'y suis pas parvenu. J'ai essayé de me renseigner sur les XYChart sur internet et la documentation mais je n'ai pas eu le temps de compléter cette tâche

Description de l'interface :

Je l'ai réalisé en amont grâce à Scenebuilder.

J'ai deux boutons radio reliés par un ToggleGroup, 4 textFields, 4 boutons et un slider. J'ai créé une échelle avec un VBox composée de rectangles de couleur à laquelle j'ai juxtaposé des labels pour les températures. J'ai, sur la fin du projet, ajouté la partie de droite composé du graphique à mon interface de base mais comme je l'ai dit précédemment, je n'ai pas eu le temps de finir cela. Je souhaitais afficher juste à coté du globe le graphe correspondant à l'année en question.

On peut zoomer et se déplacer comme on le veut autour du globe. Lorsque l'on n'a pas de données pour une zone à une année donnée, rien ne s'affiche pour ne pas obstruer les autres anomalies et ne pas perturber l'utilisateur.

Annexes :

Diagramme UML initial de la partie applicative :

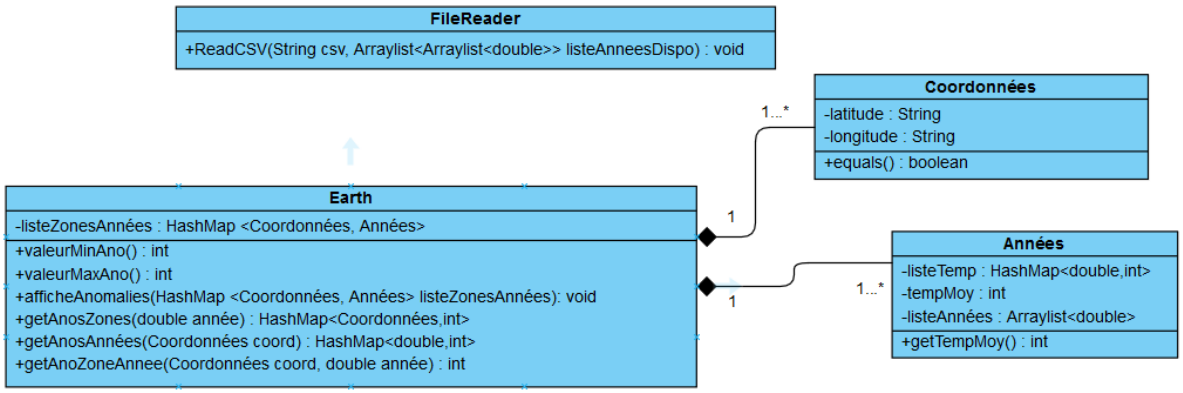


Diagramme UML final de la partie applicative :

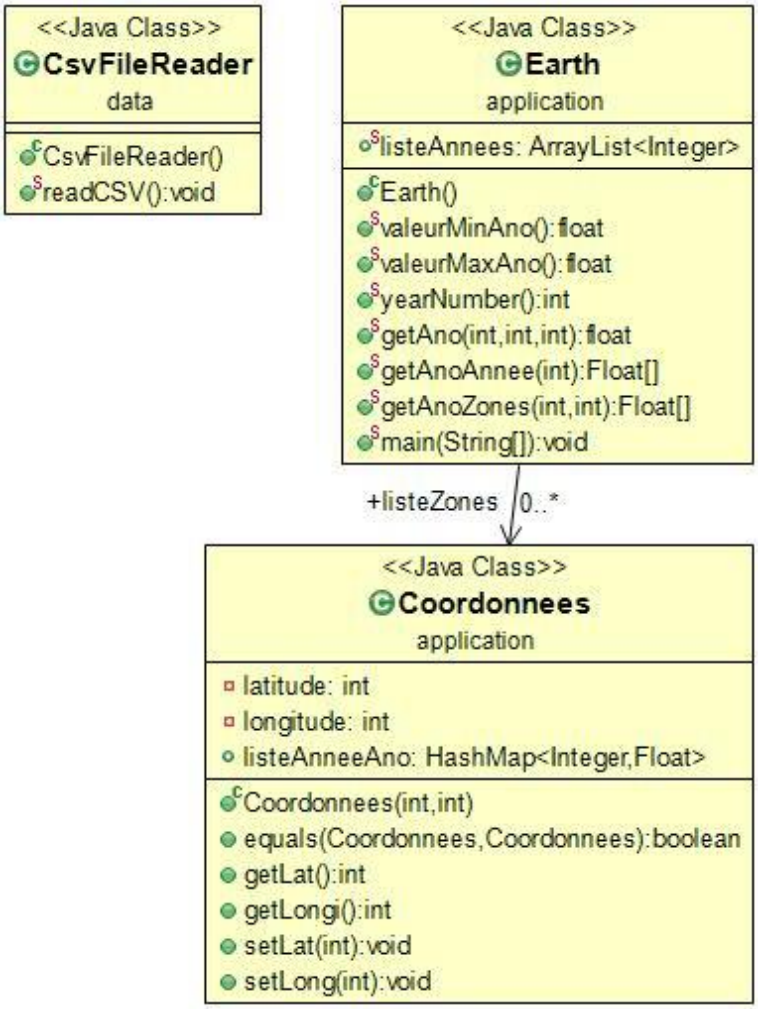
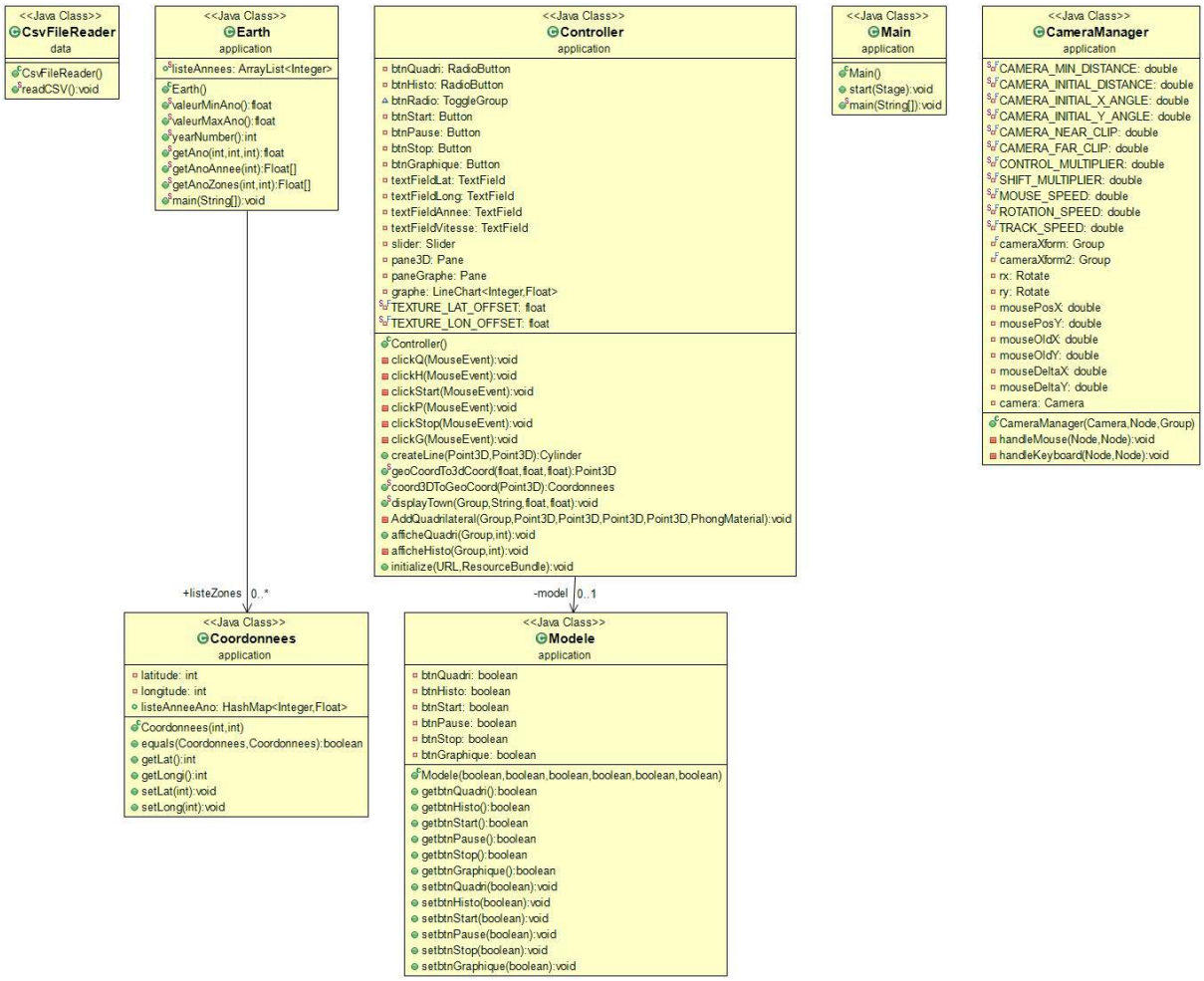
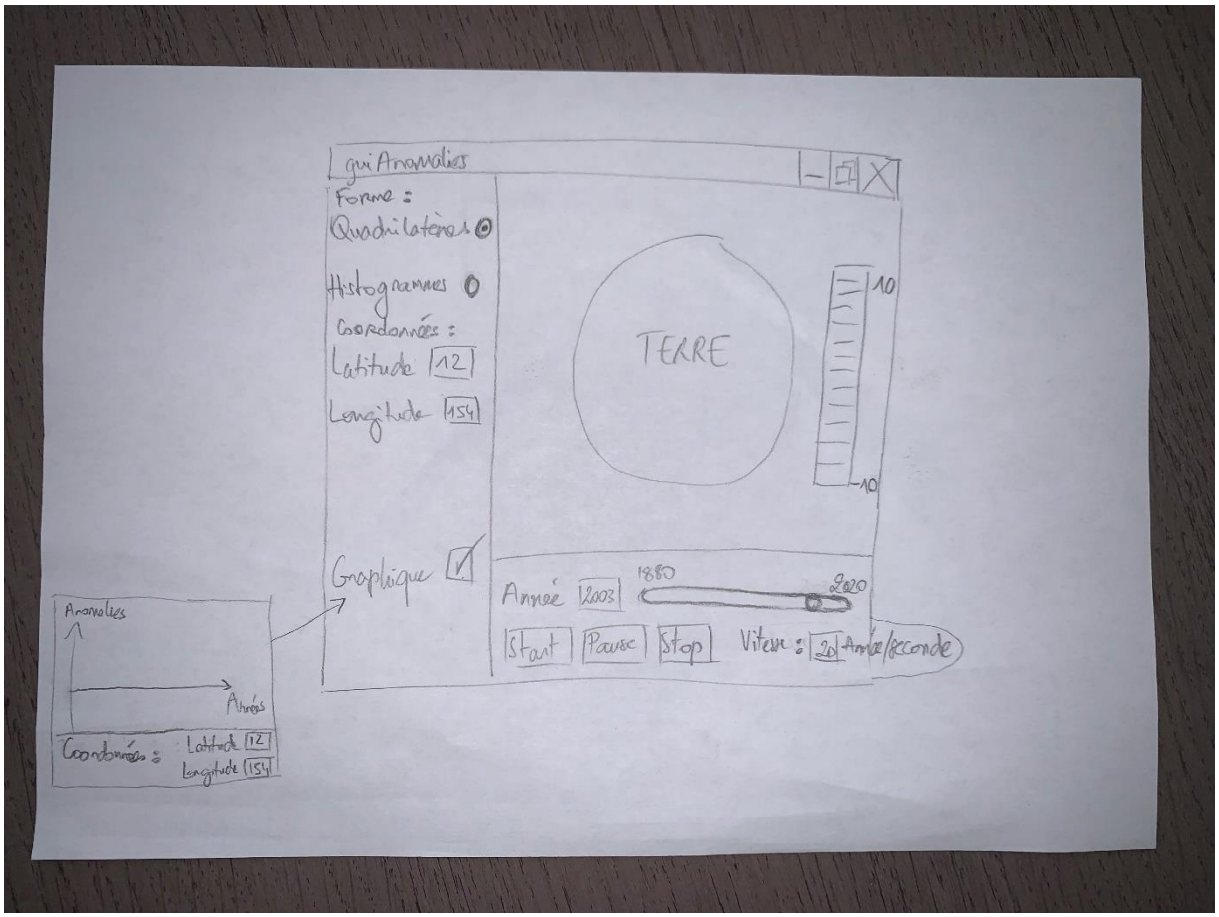


Diagramme UML final global :



Prototype papier de l'interface graphique :



Screenshot de l'application après implémentation :