Decotignie Matthieu - 329953
BOURGEOIS Thibaud - 324604

ReadMe - tCHu

Création des animations de transition lors de la prise de FaceUpCards.

- Lors de la prise d'une `faceUpCard`, la carte :
  - S'agrandit et se déplace au centre de l'écran tout en tournant.
  - Revient vers sa position initiale en rétrécissant.

- Pour se faire, plusieurs objets JavaFX nous ont été utiles. Ils ont été implémentés dans la classe `DecksViewCreator` dans notre boucle qui parcourt l'ensemble des `FACE_UP_CARDS_SLOT`.

- Ainsi, en cliquant sur une carte, grâce à `setOnMouseClicked` du node de la carte (si la carte n'est pas disable), les animations se lancent. Celles-ci sont utilisées :
  - `TranslateTransition` afin de déplacer les cartes au centre de l'écran.
  - `ScaleTransition` afin de les faire s'agrandir.
  - `RotateTransition` afin de les faire tournoyer.

- La couleur de la carte ne s'actualise pas au cours de l'animation pour le joueur qui vient de la tirer mais seulement à la fin grâce à la méthode `setOnFinished()`. Pour cela nous avons modifié le listener de `faceUpCardProperty` du slot correspondant.

Création d'une carte multicolore

- Nous avons choisi de créer une nouvelle carte multicolore (présente au nombre de 2 dans le jeu) permettant, au joueur qui l'a en sa possession, de détruire la route adverse de son choix. Les wagons ayant été utilisés dans la prise de cette route sont redonnés au joueur adverse et la route détruite repasse à un état non possédé. Ainsi la route peut dès le tour suivant être reprise par n'importe lequel des deux joueurs.

- Pour se faire, nous avons d'abord défini la nouvelle carte dans la classe énumérée `Card : MULTICOLOR`. Puis une nouvelle couleur dans la classe `Color` que nous avons spécifié dans `color.css` grâce à une image PNG ajoutée dans les ressources.

- Nous avons créé :

  - Un nouveau message dans la classe Info annonçant la destruction de la route.
  - Une méthode `withDestroyedRoute(Route route)` dans la classe `PlayerState` retournant un `PlayerState` similaire sans la route passée en argument.

    - Une méthode `withDestroyedRoute(Route route)` dans le `GameState` qui retourne un `GameState` similaire où la route passée en argument est retirée au joueur adverse et la carte multicolore est enlevée de la main du joueur actuel pour être mise dans la défausse.

- Nous avons modifié les `possibleClaimCards` de `Route` de telle sorte qu'il y ait systématiquement la carte multicolore. La méthode `canClaimRoute` de `PlayerState` a été modifiée afin que la carte multicolore ne soit pas utilisée pour s'emparer d'une route.

- En parcourant toutes les routes du jeu dans `setState` de `ObservableGameState`, si la route parcourue est possédée par le joueur adverse, que le joueur actuel possède au moins une carte multicolore, alors la `BooleanProperty` indiquant si l'on peut interagir avec la route est mise à `true`.

- Pour finir nous avons ajouté dans `Game` dans le cas de `CLAIM_ROUTE`, un envoi d'information indiquant que la route sélectionnée est détruite si `initialClaimCards` est composée d'uniquement une seule carte multicolore.
- Le nouveau `GameState` est celui retourné par la méthode `withDestroyedRoute` de `GameState`.

Affichage dans la liste des billets si certains sont complétés

- Si le billet est complété sa couleur de fond passe de rouge à vert et le symbole à la fin de son texte change de ✗ (une croix) à ✓ (un check mark).

- Nous avons créé une méthode `ticketsDone(Ticket t)` dans le `PlayerState` permettant de savoir si l'état du joueur correspondant contient le billet et si oui, s'il l'a complété. Ensuite, nous avons défini un attribut `ticketsComplete` de type `Map<Ticket,BooleanProperty>` dans `ObservableGameState`. A l'appel de la méthode `setState`, on parcourt tous les billets du joueur et on met la `BooleanProperty` correspondant au ticket à la valeur que retourne `ticketsDone`.

- Nous avons modifié la méthode `setCellFactory` de la `ListView,` présente dans la classe `DecksViewCreator` correspondant aux billets, de telle sorte que la `textProperty` soit liée à la `BooleanProperty` du billet à l'aide des méthodes `Bindings : .when() .then()` et `.otherwise(`. Ainsi, elle affiche le texte désiré en fonction de si le billet est complété ou non. Enfin, nous avons fait de même avec la `styleProperty` avec le code css décrivant la couleur du fond.

Affichage en continu du chemin le plus long

- Au cours de la partie, le chemin le plus long est mis en avant. La couleur et la taille des cercles de chaque case de ses routes changent : les cercles sont grossis et pleins.
- Nous avons ajouté un attribut de type `Map<Route,BooleanProperty>` à `ObservableGameState`. Dans la méthode `setState` de `ObservableGameState` on obtient alors le chemin le plus long de chaque joueur. Ensuite, on compare leur longueur afin de déterminer lequel afficher. Dans le cas où les deux chemins sont de même longueur, les deux sont affichés comme étant les plus longs.
- Enfin on parcourt toutes les routes du jeu. Si la route parcourue est contenue dans le plus long chemin, sa `BooleanProperty` est mise à `true`. Dans la classe `MapViewCreator` on a lié la `radiusProperty` ainsi que la `fillProperty` des cercles contenus dans les cases des routes à la `BooleanProperty` de la route en question, afin que le rayon augmente et que la couleur de remplissage change de blanche à grise lorsque la propriété est vraie, et inversement lorsqu'elle est fausse.