Rocher Jean-Francois (316766) Rousselot Evgueni (320195)

Rapport extension Tchu

Extension 1 : Affichage du nombre total de points

- Amélioration pour l'utilisateur :

  En affichant le nombre total de points en temps réel cela permet à l'utilisateur de savoir comment il se situe précisément dans la partie.

- Mise en oeuvre Java :

  Création d'une **Property** dans **ObservableGameState** qui est mise à jour par l'appel de **myPlayersState.finalPoints()**.

  De plus, on effectue en binding avec un **Text** du **DecksViewCreator** pour pouvoir l'afficher.

Extension 2 : Affichage du temps et attribution d'un nouveau bonus pour le joueur le plus rapide

- Amélioration pour l'utilisateur :

  Le temps pris par chaque joueur pour jouer est affiché et mis à jour à chaque fin d'action. Cela permet d'inciter à jouer une partie plus rapidement, car un bonus de 20 points est attribué à celui qui aura finis la partie en jouant le plus rapidement .

- Mise en oeuvre Java :

  Comme pour l'extension 1, on a rajouté une **Property** dans **ObservableGameState** et un binding lié à un **Text** du **DecksViewCreator** pour pouvoir l'afficher.

  On a dû créer une variable **time** dans la classe **PlayerState** stockant le temps mis par le **Player**. Pour cela il a fallu modifier plusieurs méthodes de **PlayerState**, **GameState** et **Game** pour correctement mettre à jours l'attribut time et être en accord avec le nouveau constructeur de **PlayerState**.

  Pour connaitre le temps écoulé, on fait appel à la méthode **System.nanoTime()** et avec la méthode **getTimePlayer(...)** de **Game** on met ainsi correctement à jours le temps écoulé pour chaque joueur dans l'attribut **timeMap** de **Game**.

  Une constante et un texte pour l'affichage du bonus de temps on aussi été rajoutés dans les classes **Constant** et **StringsFr**.

  L'attribut **SERDE_PLAYER_STATE** dans **Serdes** a été modifié pour pouvoir communiquer le temps.

Extension 3 : Ajout d'une animation lors du tirage de cartes

- Amélioration pour l'utilisateur :

  Pour une meilleure expérience visuelle, une animation a été ajoutée lors du tirage de cartes. Cela permet aussi de mieux voir qu'elle carte a été tirée de la pioche. Les animations ont été conçues pour la taille par défaut de la fenêtre de Tchu, par conséquent, l'agrandir décalera les animations.

- Mise en oeuvre Java :

  Dans **GraphicalPlayer**, la méthode **animation(…)** est appelée à chaque fois qu'une carte est piochée. Elle crée un **StackPane** qui a l'apparence de la carte piochée, et le positionne en fonction de l'attribut **slot**.

  La méthode lance ensuite une animation de type **TranslateTransition**, qui déplace le pane au dessus de la position que la carte prendra dans la main du joueur. A la fin de celle-ci, le pane est enlevé des noeuds du **BorderPane** et la méthode **onDrawCard(…)** est appelée.

  La **topDeckCard** ne peut pas être récupérée à partir d'un **PublicGameState**, à l'inverse d'une **faceUpCard** qui est stockée dans une propriété de **ObservableGameState**. Cela posa un problème pour son animation.

  La **topDeckCard** est donc récupérée avec la méthode **updateState(…)** de **Player** qui a été modifiée notamment avec l'ajout de l'attribut **Card topCard**.

  Nous avons donc modifié en conséquence toutes les méthodes liées, ainsi que les sérialisations et désérialisations, pour placer cette carte dans un propriété de **ObservableGameState**, que nous récupérons avec un getter dans **animation(…)**.

  Toute les grandeurs associées ont été mises dans **Constantes**.

Extension 4 : Mise en évidence des gares aux extrémités d'un billet

- Amélioration pour l'utilisateur :

  En sélectionnant l'un de ses billets, le joueur peut voir apparaitre sur la carte les deux gares aux extrémités à l'aide d'images. Les pays ne sont pas mis en évidence, ils sont déjà biens visibles.

- Mise en oeuvre Java :

  Nous créons dans **MapViewCreator** un cercle par stations n'étant pas un pays, puis nous lui associons une image de **Ressources** et une position, grâce à une classe et un id que nous avons préalablement définis dans le fichier **map.css**.

  Les lignes css contenant les coordonnés de chaque gares ont été générées par un programme java (voir classe **Annexe** du **package gui**), où les coordonnées étaient récupérées par la position du clique.

  Dans **ObservableGameState** nous avons créé une liste de **SimpleBooleanProperty**, une par station, qui sera ensuite bind à la **visibleProperty** du cercle associé à cette station dans **MapViewCreator**.

  Ces propriétés sont mises à jour par une méthode dans **ObservableGameState** **updateSelectedStation**(**ObservableList<Ticket> ticketList**) qui set à true uniquement les propriétés liées aux stations aux extrémités des tickets de **ticketList**.

  Cette méthode est alors appelée par un listener d'**ObservableList** (**ListChangeListener**) dans **createHandView(…)**, associé à la liste observable des tickets sélectionnés.