Rapport

tCHu - étape 12

Ménélik Nouvellon - Albert Troussard

(lien ressources: https://drive.google.com/file/d/1h_AhQRilD_ra9_xVVCp95tUq_oNb6m6w/view?usp=sharing)

Améliorations/extensions

- **menu (Classe `Main`) :** Ajout d'un menu permettant au joueur de facilement démarrer sa partie que ce soit pour le client ou pour le serveur. De plus, sur cette page est affichée l'adresse Ip du joueur pour lui permettre de facilement la transmettre à son adversaire (il a la possibilité de cliquer sur l'adresse Ip pour la copier)
- **aéroports et avions :** des aéroports ont été ajoutés dans 4 villes aux extrémités de la carte : Delémont, Genève, Brusio et Saint-Gall. Les joueurs ont la possibilité de relier deux de ces villes à l'aide d'une carte Avion (carte spéciale que l'on a ajoutée dans le jeu et n'existe qu'en 2 exemplaires dans tout le jeu). Lorsque une route aérienne a été prise, toutes les autres sont bloquées pour le reste de la partie. Quand on tente de s'emparer d'une route aérienne, un nombre aléatoire et choisi entre 0 et 3 (inclus) et pour réellement s'emparer de la route, le joueur devra payer comme coût supplémentaire ce nombre en carte locomotive uniquement
- **outil de dessin :** à tout moment lors de la partie, le joueur a la possibilité de dessiner sur la carte que ce soit pour esquisser une stratégie ou pour s'amuser en attendant que l'autre joueur joue. Sont également disponibles une gomme, un bouton reset pour reinitialiser tous ses dessins, un ColorChooser pour changer de couleur et un slider pour choisir la taille du stylo
- **mise en avant des stations du billet choisi :** lorsque l'on clique sur un de nos billets, un cercle rose apparaît sur chacune des stations impliquées dans le billet pour permettre au joueur de mieux visualiser l'emplacement des stations concernées
- **joueur actuel :** pour permettre facilement à un joueur de savoir si c'est à lui de jouer, nous avons mis à disposition un cercle accompagné d'un court texte indiquants si c'est au tour du joueur (rouge si il doit patienter et vert si c'est à lui de jouer)
- **animations et améliorations graphiques** :
    - lorsque le joueur peut s'en emparer et qu'il passe la souris sur une carte face visible, celle-ci s'agrandit légèrement à l'instar des routes qui peuvent être prises par le joueur
    - lorsqu'un joueur prends une route, une animation se produit sur tous les rectangles de la route en question
    - lorsque le joueur s'empare d'une route aérienne, les avions situés au départ et à l'arrivée de la route vont en direction de la station opposée
    - ajout d'icônes pour le programme dans la barre des tâches

- **sons:** au début du jeu, lorsque le joueur s'empare d'une route terrestre, d'un route aérienne ou lorsque il pioche une carte de la pioche, un son est produit en fonction du contexte

Mise en oeuvre Java (et CSS)

- **menu:** Nous avons créé une classe `Menu` qui hérite de `Application`, composée d'une image d'arrière-plan, du logo de notre jeu ainsi que de différents boutons auxquels nous avons ajouté des `Listener`. Nous avons arrangé le tout dans différentes `VBox` et `Group` avec des `setTranslateX()` et `setTranslateY()`
- **aéroports et avions:** Nous avons ajoutés une carte *Avion*, de couleur `null` ainsi que qu'un `Level (SKY)` et de nouvelles routes pour chaque liaison possibles. Pour faire en sorte que uniquement une route aérienne ne puisse être prise, nous utilisons une `booleanProperty` qui devient faux lorsqu'un joueur prends une de ces routes, bloquant ainsi les autres
- **outil de dessin:** Un `Canvas` de la taille de la carte est créé et est rendu actif uniquement quand le bouton de dessin ou de gomme est pressé. Quand le joueur clique ou maintient son clique dans la zone de dessin, un trait est ajouté au canvas grâce aux méthodes `lineTo` et `stroke` (et `beginPath` pour réinitialiser le trait). La taille du crayon est liée à la propriété du `Slider` et sa couleur à celle du `ColorPicker`. Le bouton *Reset* appelle la méthode `clearRect` sur l'ensemble du canvas, ce qui efface donc tous les précédents dessins. La *Gomme* fait de même mais sur zone plus réduite.
- **mise en avant des stations du billet choisi :** D'abord, nous avons créé un cercle rose pour chaque station auquel nous avons associé leur id respectif, placé ensuite à la bonne position au moyen du fichier *map.css*. Puis nous avons mis dans `DeckViewCreator` un `ObjectProperty<Ticket>` qui contient le ticket sélectionné de la liste de tickets du joueur contenu dans la vue de main. Cette propriété est récupérée dans `GraphicalPlayer` qui se charge de rendre visible uniquement les cercles pour les stations impliquées dans le billet.
- **joueur actuel:** Après avoir créé dans `ObservableGameState` un `ObjectProperty<PlayerId>` qui nous donne l'id du joueur courant, nous l'utilisons dans `DeckViewCreator` en lui ajoutant un `Listener` faisant en sorte que si l'id de ce dernier est le même que l'id du propriétaire du `observableGameState` courant, alors un texte contextuel s'affiche et vice-versa
- **animations:** Pour toutes les animations sauf pour celle des cartes face visible, nous avons utilisé des concepts Java tel que `Timeline` qui permet d'avoir un fil temporel décrivant les actions à faire ou encore `TranslateTransition` qui permet une transition d'un point (x_1, y_1) vers un autre point (x_2, y_2).
- **sons:** Nous avons utilisé le concept de `Clip` et de `AudioInputStream`