Fonctions JS Baroudeur des trains

Toutes les fonction décrites se trouve dans ressources/js/map.js
Les fonctions sont toutes appelées après le chargement du DOM

A partir de la fonction du 4. , on utilise un tableau “**playersInfo**“ regroupant les infos des joueurs, il est construit sous cette forme (exemple) :


```
{
        "player1": {
            "points": 6,
            "color": "blue",
            "rides": ["plymouth_bristol", "douai_bruxelles", "nancy_dijon_2"]
        },
        "player2": {
            "points": 4,
            "color": "green",
            "rides": ["clermont_bordeaux", "nancy_dijon_1"]
        },
        "player3": {
            "points": 4,
            "color": "red",
            "rides": ["ajaccio_barcelone"]
        },
        "player4": {
            "points": 4,
            "color": "yellow",
            "rides": []
        }
    }
```


PS : là les points sont pas bien calculés c’est un exemple (j’ai dissocié les points des trajets pour qu’on ait pas de pb quand on devra calculer les points finaux avec les objectifs et les gares etc…)

1. createRails(map, coordinates) : void

Localisation : ligne 1

Créé dans path svg pour chaque rail et leur donne deux classe et un id : une classe “**rail**” et une autre ayant le même nom que le trajet auquel appartient le rail. L’id est le même que le nom de la deuxième classe mais on rajoute à la fin un “**-**” suivi du numéro du rail dans le trajet.

Exemple, pour le trajet entre Bristol et Londres, composé de 3 rails, on aura dans le svg :
<path id="bristol_londres-0" fill="transparent" class="bristol_londres rail" d="..."></path>
<path id="bristol_londres-1" fill="transparent" class="bristol_londres rail" d="..."></path>
<path id="bristol_londres-2" fill="transparent" class="bristol_londres rail" d="..."></path>

2. addIntercationWithRails(map) : void

Localisation : ligne 26

Ajoute des events listeners sur les rails (à compléter)

3. displayDestinationGoal(map, coordinates) : void

Localisation : ligne 50

But : afficher les objectifs de destination du joueur

Komen sa march : ça va prendre tous les éléments du DOM qui ont la classe “destination” et regarder leurs ids qui seront des objectifs.
(ex : limoges_milan)
La fonction met 2 events listeners sur ces objets (mouseover et mouseout) pour que la destination s’affiche quand on passe la souris sur l’objet et disparaisse quand on l’enlève.

A penser : Mettre des objets avec la class destination sur le DOM pour chaque joueur

4. playerPositionSameScore(playersInfo, player) : [int, int]

Localisation : ligne 86

Fonction qui sert dans la fonction d'après (lis pas ce qu'elle fait si t'as pas le temps)
Prend en entrée les infos des joueurs ainsi que le nom d'un joueur
Renvoie une liste de 2 éléments, le deuxième est le nombre de joueurs ayant le même score que le joueur passé en entrée (modulo 100), et le premier les la position du joueur dans l'objet "**playersInfo**" parmi les joueurs ayant le même score.

exemple, dans l'objet "**playersInfo**" ci-dessus, en page 1, on aurait ceci :
playerPositionSameScore(playersInfo, "player1") → [1, 1]
playerPositionSameScore(playersInfo, "player2") → [1, 3]
playerPositionSameScore(playersInfo, "player3") → [2, 3]
playerPositionSameScore(playersInfo, "player4") → [3, 3]

5. displayScores(playersInfo, map, coordinates) : void

Localisation : ligne 104

Affiche des pions de la couleur du joueur sur les cases bleues autour du plateau, la fonction prend les scores (modulo 100) dans la catégorie "**points**" de chaque joueur. Elle permet de prendre en compte le cas où plusieurs joueurs sont sur la même case grâce à la fonction playerPositionSameScore.

6. displayRides(playerList, map) : void

Localisation : ligne 137

Parcours les trajets posés par les joueurs et remplit les rails de chaque trajet de la couleur du joueur. (Chaque rail d'un trajet X sur le svg a la classe X) donc il suffit de prendre tous les éléments ayant la classe du même nom que le trajet.