Rapport Projet Rover

Pour notre projet, nous avons décidé de créer un petit jeu avec un Rover qui se déplace sur mars. Il doit naviguer sur la carte et récupérer des pièces détacher pour réparer une antenne cassée. On perd quand la batterie du rover est vide ou bien s’il y a plus de temps, et on gagne quand l’antenne est réparé dans le temps imparti.

# Partie Julien BOUGUES :

Durant le développement de notre jeu, j'ai pris en charge plusieurs aspects essentiels à l'interaction utilisateur et à la dynamique de jeu.

## Commandes :

Ce menu permet au joueur de connaître les touches nécessaires pour déplacer le rover et interagir avec les éléments du jeu. Il a été réaliser sur SceneBuilder

## Map :

Ensuite, j'ai entièrement développé la map du jeu. La carte a été implémentée à l'aide d'un grand Canvas sur lequel est dessinée l'image de fond représentant Mars. Pour afficher le rover, j'ai codé un mécanisme de caméra centré sur ce dernier, permettant à la vue principale de se déplacer en synchronisation avec les mouvements du rover. Le déplacement du rover est géré par la capture d'événements clavier, en particulier les touches directionnelles dans le controllerMap.  
  
Batterie :

Pour ajouter du réalisme et une contrainte au joueur, j'ai conçu une gestion complète de la batterie. La batterie est représentée visuellement par une ProgressBar qui diminue linéairement au fil du temps. Le niveau de la batterie décroît constamment sauf quand le rover se trouve dans un rayon précis autour d'une base, auquel cas la batterie se recharge progressivement. Pour réaliser ce mécanisme, j'ai utilisé des méthodes spécifiques (tick() pour la dépense et recharge() pour le rechargement) intégrées au model ModelCar. Cette structure a permis de séparer clairement la logique métier de l'affichage visuel.

## Skins :

J'ai également intégré différents skins pour personnaliser le rover. Ces skins, récupérés sur Internet, sont stockés sous forme d'images dans les ressources du projet. Le changement se fait ensuite dans le menu personnalisation fait par Raphaël.

## Pause :

Enfin, j'ai mis en place le menu "Pause" accessible à tout moment durant la partie par une touche dédiée (Échap). Ce menu arrête complètement la progression du jeu, notamment en mettant en pause l'AnimationTimer et les timelines responsables du temps et de la batterie, garantissant ainsi l'intégrité des mécanismes du jeu pendant cette pause. Les boutons de ce menu permettent ensuite de reprendre la partie exactement à l'état dans lequel elle avait été suspendue ou de naviguer vers d'autres menus.

# Partie Raphaël CHARTON :

## Menu :

Pour le menu, j’ai d’abord fait le fichier en FXML avec Scene Builder, avec des buttons, et des radiobuttons. Ensuite j’ai fait le controller qui permet d’afficher les différents onglets comme Personnalisation, Jouer et Quitter. De plus il y a la difficulté disponible avec les radiobuttons.

## Personnalisation :

Comme pour l’onglet menu j’ai décidé de le faire sur Scene Builder pour pouvoir utiliser le container ImageView pour pouvoir faire défiler les images des différents skins (avec une flèche de chaque côté). Il y aussi un bouton Retour (qui retourne sur l’onglet précédent) et un bouton Valider (qui sauvegarde le skin choisit).

## Fenêtre de jeu et minuteur :

J’ai ajouté la progressBar et l’inventaire sur SceneBuilder avec des canvas et des lines. Ensuite j’ai ajouté le minuteur en haut à droite avec le temps qui défile.

## Fenêtres de fin :

Pour les fenêtres de fin je les ai aussi créé sur Scene Builder. Elles sont appelées lorsque la batterie est vide, que le chronomètre est à 0 ou que tous les objets sont mis sur l’antenne. Sur ces fenêtres on peut rejouer, personnaliser le véhicule ou quitter.

## Mise en place des items sur la map :

Emplacement des items aléatoires avec certaines distances en fonction de la difficulté. Ensuite affichage sur la minimap en rouge (affichage ou non des objets en fonction s’ils ont déjà servi à réparer l’antenne ou s’ils sont dans l’inventaire). Emplacement de l’antenne en fonction de la difficulté. Changement de skin de l’antenne en fonction de l’avancé des réparations et affichage en bleu sur la minimap.

## Drag and drop :

Pour le drag and drop je vérifie si le clic est fait sur un objet et ensuite s’il est fait sur un objet alors je récupère ses coordonnées dynamiquement pour qu’il s’affiche à l’endroit où est la souris puis lors du release je vérifie si l’objet est lâché sur l’antenne (si oui je fais disparaitre l’objet et je passe à l’étape de réparation d’après, sinon il est toujours visible).

## Inventaire :

J’ai eu du mal à le faire car je ne savais pas comment j’allais sortir les objets de l’inventaire. J’ai donc décidé de faire des canvas et lorsque je clique sur un canvas et que l’objet est dans cet emplacement alors il est drop près de la case. Ensuite je peux faire le drag and drop.

## Mouvement du véhicule :

Je récupère sur quelle flèche on clique pour orienter le véhicule dans la bonne direction.