Henrot Adrien Groupe 3 Décembre 2021

Rapport du Projet d'IPO

Introduction :

Ce projet a été réalisé en binôme. Pour les parties « obligatoires » (parties 1,2 et 3) chacun a effectué les taches demandées de son côté durant les vacances d'octobre avec quelques interactions via Discord si l'un ou l'autre se retrouvait coincé ou avait besoin d'une précision. Puis par la suite nous avons menés à bien ce projet en nous voyant régulièrement pour échanger sur ce dernier ou encore une fois en échangeant des informations sur Discord. Nous avons eu certes quelques difficultés lors de la programmation mais aucun problème n'est resté non résolu.

1) Partie 1 du projet : La grenouille

Aucune difficulté pour implémenter cette partie n'a été relevé de notre côté. Les fonctions de base ont été faites en une après midi.

2) Partie 2 : L'environnement : Suites de routes

Difficultés :

- Comprendre comment faire bouger les voitures comme sur le .jar servant d'exemple.
- Faire une fonction Update devant s'appliquer à trois classes différentes pour faire en sorte que la partie fonctionne. Notamment pour faire bouger les voitures avec les ticks, cela a nécessité de la réflexion sur le sens de cette fonction. En effet cette fonction est basée sur le temps d'exécution du programme en lui-même.

3) Partie 3 : Jeu Infini

C'est dans cette partie que nous avons rencontré le plus de difficulté (d'un point de vue de logique). Nous avons d'abord implémenté une méthode infinie qui rajoutait des lignes en haut du tableau et supprimait celle sur lesquelles la grenouille passait. Mais nous nous sommes rendus comptes qu'en faisait ça il était impossible pour la grenouille de revenir en arrière ce qui n'était pas très convenable pour un jeu n'ayant pas tant de fonctionnalités que ça.
Nous en avons donc déduit que :

- la grenouille doit monter mais ne doit pas bouger graphiquement
- la taille du tableau de routes doit augmenter en gardant la fenêtre graphique stable. Les anciennes routes seront donc gardées jusqu'à un certain point.
- si la grenouille monte, il faut qu'une nouvelle route se crée en haut et que le tout descende et ce à chaque déplacement de la grenouille.

Pour résoudre ses problèmes nous avons donc modifier la déclaration de Element lié à la grenouille. Ensuite nous avons dans la fonction move de frog nous avons rajouter une fonction permettant de rajouter des lignes.

Le dernier point a été le plus dur à résoudre. Mais en jouant sur l'affichage graphique des voitures en les faisant changer d'ordonnée notre tableau contenait toujours les anciennes routes.

4) Éléments complémentaires :

1) Le Timer

Étant donné que java propose une fonction pour récupérer l'heure de de la machine sur laquelle le programme tourne, nous avons pu assez facilement, grâce a une classe chrono implémentée dans Game faisant la soustraction entre l'heure à laquelle le jeu à démarrer et l'heure à laquelle le joueur perd la partie.

2) Cases Spéciales

Nous avons implémenté trois cases spéciales qui ne nous ont pas vraiment posées de problèmes.

- Une case arbre que la grenouille doit contourner pour continuer d'avancer
- Une case Toile qui entrave les mouvements de la grenouille : le joueur doit appuyer trois fois sur le bouton de déplacement pour que cela fonctionne.
- Une case n'étant pas un obstacle mais un bonus : la case pièce, si la grenouilles passe dessus le joueur gagnes un bonus d'un point.

3) Rivières et rondins

Pour cette partie la première étape a été de tester de créer une classe Rivière et une classe rondin ayant des propriétés similaires a Lanes et Car, puis les superposer à notre tableau déjà existant mais cela ne marchait pas, des erreurs types ArrayOutOfBands apparaissaient dans le terminal. Nous avons donc réfléchi puis décider que nous pouvions utiliser les classes déjà existantes pour créer nos « objets » rivières et rondin. Il suffisait de jouer avec les paramètres de ses derniers pour les adapter à nos besoins. Il nous a ensuite fallu changer dans les paramètres de la grenouille quelques propriétés pour faire en sorte qu'elle considère les rondins comme safe et que ceux-ci puissent la faire bouger de gauche à droite. (Nous avons appliqué les propriétés des rondis a la grenouille quand celle-ci est dessus)

5) Bonus

1) Jeu a deux joueurs

Cette partie n'as pas été spécialement compliqué, il s'agissait surtout de lier une deuxième grenouille ayant les mêmes propriétés que la première a notre partie en changeant les touches de déplacement ainsi que et en faisant en sorte que note classe lane prenne la deuxième grenouille en compte. Il a bien sûr aussi fallu changer les endroits d'apparition des grenouilles en début de partie.

2) Meilleur rendu graphique

Pour avoir un jeu avec des graphisme, nous avons utilisés des sprites trouvés sur internet (nous avons essayé d'en faire à la main mais nos talents de graphiste n'étaient pas au rendez-vous). Grace à la méthode paintComponent() de FroggerGraphics nous avons pu grâce à la classe ImageIcon lié les images à nos objets . Nous avons rencontré quelques difficultés pour les objets ayant une taille supérieur à la taille de la case mais nous avons résolu ce problème en liant l'image non pas à l'objet total mais a sa première case uniquement.

3) Nos Idées !

Dans cette partie nous avons décidés d'implémenter tous les éléments que nous avions codés pour en faire un « vrai jeu ».
Nous avons donc implémenté des boutons permettant de choisir :
- le mode de jeu (infini ou limité)
- le nombre de joueurs (1 ou 2)
- la difficulté (en changeant la densité des voitures traversant les lanes)
- un bouton de restart sur le endGameScreen après être mort
- un bouton Home de retour au menu principal.

Cela n'a pas été d'une complexité débordante, seulement le bouton de restart est celui qui a été le plus complexe. Recommencer une partie voyant l'implémentation de nos classes de base était un bon challenge.

Enfin nous avons implémenté des bruitages quand la grenouille se déplace ainsi qu'une musique de fond.