Jacobo Ruiz Ocampo
Alexis Delplace

Rapport : Projet IPO “Frogger”

Implémentation

	Alexis	Jacobo
Partie 1	✔️	✔️
Partie 2	✔️	✔️
Partie 3	✔️	✔️
Partie 4 : Timer	✔️	❌
Partie 4 : Cases spéciales	M	✔️
Bonus : Sauvegarde des meilleurs temps et score	M	✔️
Bonus : 2 joueurs	✔️	❌
Bonus : Rendu graphique	✔️	❌
Bonus : Menu principal	✔️	❌

Légende : ✔️ → Coder | ❌ → Pas coder | M → Modification

Workflow

Comme vu dans la section “implémentation”, nous avons parfois tous deux implémenté les mêmes parties / fonctionnalités. Lorsque c’était le cas, nous avons utilisé l'outil de comparaison de code intégré à Intelij IDEA, qui permet de comparer sa branche avec la branche de l’autre ou la branche master.
Pour ce qui est du reste, même si Jacobo a implémenté moins de fonctionnalités (bonus), il a examiné et compris la totalité du code.

Problèmes rencontrés et solutions trouvés

Situation 1:

Jacobo a tenté d'implémenter les “lignes d'eau”. Dans un premier temps, il a essayé d’étendre la classe lane pour créer une classe “river” mais cela a causé beaucoup de problèmes d’incompatibilité lors de l’ajout d’une rivière à la liste de lanes aussi appelé “lanes” dans la classe “environment”.

Après plusieurs essais de debuggage il a essayé de tout refaire depuis le début en modifiant le type de la liste “lanes” vers un nouveau type “lignes” beaucoup plus générique qui serait étendu par la classe “lane“ et ”river” mais ceci n’a pas éliminé les problèmes de compatibilité et ça

n'était pas plus simple que ce qu'il avait essayé de faire dans un premier temps. L'erreur ici, a clairement été d'avoir supprimé la première version de son code.
Pressé par le temps, il a décidé de laisser tomber cette fonctionnalité non nécessaire au projet pour se concentrer sur les parties plus importantes.

Situation 2:
Pour développer la partie p+1 on devait attendre que l'autre personne finisse la partie p et la push pour pouvoir la pull, la modifier et ainsi la push. C'est ainsi que l'on a découvert la fonctionnalité "merge".

Situation 3:
Alexis voulait améliorer les "endGameScreen" en ajoutant un bouton "rejouer" et "menu principal", le problème étant que le programme n'était pas conçu dans ce sens et qu'il n'a pas trouvé comment "réinitialiser" les classes "Froggergraphic" et "Game".

Il pense qu'il aurait fallu mettre toute la partie de la classe "Main" qui gère le jeu dans une autre classe et appeler une méthode de cette classe afin de relancer le jeu, mais cette modification aurait pris beaucoup de temps, et puis il aurait fallu recréer un partie dans la même fenêtre, à cause de ces différents problèmes et par manque de temps, il a abandonné l'idée.

Situation 4:
Le fait d'utiliser des interfaces nous force à déclarer des fonctions que l'on utilise pas dans les classes qui implémentent cette interface. C'est ainsi que dans notre classe Frog, nous avons des méthodes issues de FrogInf qui ne sont pas utilisées. Il y a sûrement une meilleure manière de faire.

Suite (ce que l'on aurait fait si on avait plus de temps)

- Modification du squelette du programme afin de pouvoir relancer un partie
- Un mode 2 joueur pour le mode de jeu infini, ce qui implique un gestion particulière de l'affichage pour que l'affichage suive la grenouille en tête et si l'une des grenouille sort de l'affichage elle a perdu