ENSTA Bretagne

Projet Frogger

Programmation Orientée Objet - JAVA

Emile Berteloot - SNS
Corentin Goetghebeur - SNS
Rozenn Jézégou - SNS
04/11/2021

Table des matières

1 - Introduction

Le jeu Frogger est un jeu d'arcade développé dans les années 80 par Konami. Ce jeu consiste à faire évoluer une grenouille depuis une route jusqu'à sa maison. Pour cela elle doit franchir une route sur lesquelles roulent des voitures de différentes tailles et de différentes vitesses, puis elle doit traverser une rivière en sautant sur les rondins de bois. Si la grenouille tombe dans la rivière ou touche une voiture, elle meurt. Le jeu permet également de gagner des points bonus si la grenouille mange des mouches sur son passage. Le joueur dirige la grenouille à l'aide des touches de son clavier.

Dans le cadre de ce projet, notre objectif est de développer un jeu inspiré de Frogger avec le langage JAVA et la Programmation Orientée Objet (POO).

2 - Cahier des charges

2.1 - Objectifs :

L'objectif est de créer une version du jeu Frogger. Ce jeu consiste à faire évoluer le joueur, représenté par une grenouille, dans un environnement constitué de plusieurs routes sur lesquelles roulent des voitures. Le joueur gagne s'il parvient à traverser les routes sans toucher les différents obstacles.

2.2 - Ressources :

- Equipe de trois étudiants ;
- Java ;
- Java FX ;
- Internet ;
- POO courses.

2.3 - Contraintes techniques :

Première version de jeu

La structure de l'application :

L'application doit

- Faire appel à la programmation orientée objet ;
- Contenir un package regroupant la classe représentant le jeu, la classe représentant le menu et la classe Main permettant de lancer l'application ;
- Contenir un package regroupant les classes des différents objets : Frog, Lane, Directions, Road…

La grenouille :

- Bouge si le joueur appuis sur les touches flèches ou les touches QZSD.
- Commence le jeu en bas de la fenêtre de jeu ;
- Gagne si elle parvient à traverser toute la fenêtre de jeu ;
- Meurt si elle touche une voiture ou un obstacle ;
- A une taille de 40x40 pour une fenêtre de 1200x600.

L'environnement :

- Contient une route avec plusieurs voies ;

- Sur chaque voie, des voitures roulent de la gauche vers la droite ;
- Les voitures ont différentes tailles et différentes vitesses ;
- Les voitures ont une position de départ différentes les unes des autres ;

Version améliorée du jeu

Cette version n'est pas atteinte, cependant des pistes ont été envisagées.

Le jeu :

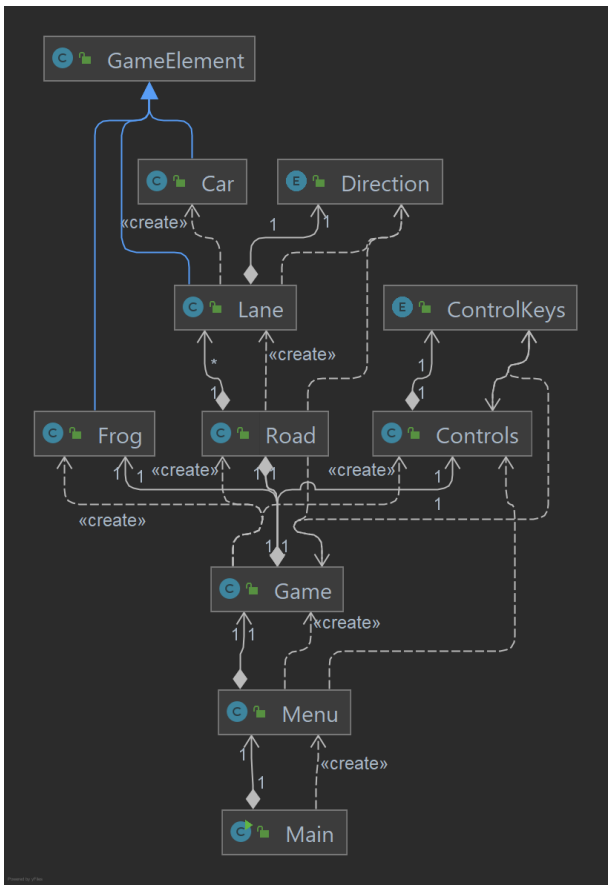
- Le jeu se lance via un menu ;
- Le temps de jeu s'affiche lorsque le joueur perd ;
- Le jeu devient infini, le rivage n'est jamais atteint, le joueur a pour but de faire évoluer la grenouille le plus longtemps possible ;
- Le jeu peut se jouer à deux avec les touches flèches et les touches ZQSD.

L'environnement :

- Des pièges sont placés sur les routes, si la grenouille les touche, elle meurt.
- Les voitures peuvent se déplacer dans les deux sens : de la gauche vers la droite et de la droite vers la gauche ;
- Un seul sens de circulation est possible par voies ;

3 - Architecture du logiciel

3.1 – Diagramme



Légende :

: Classe

: Héritage entre les classes

: Appel d'une méthode

: Création d'un objet

Figure 1 : Diagramme de structure du projet

Le jeu se décompose en deux packages : ***game.frogger2*** et ***util***. Le diagramme de classe montre comment le jeu est structuré dans ces deux packages.

3.2 – Description des classes :

Package *game.frogger2*

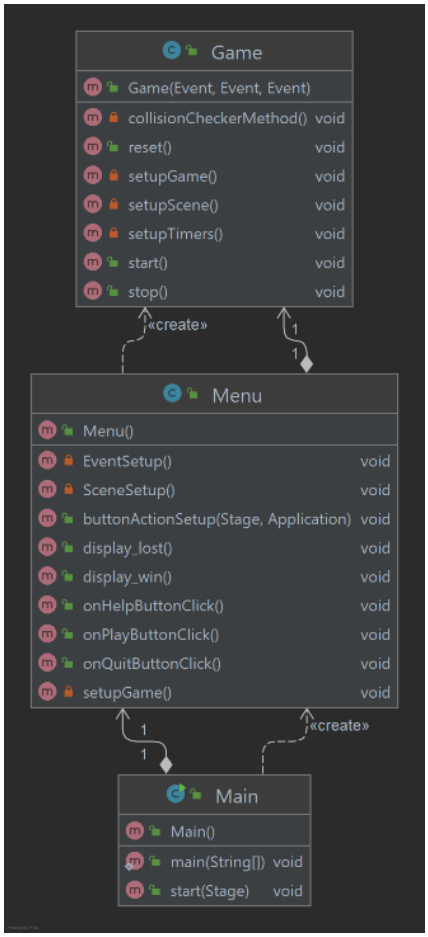
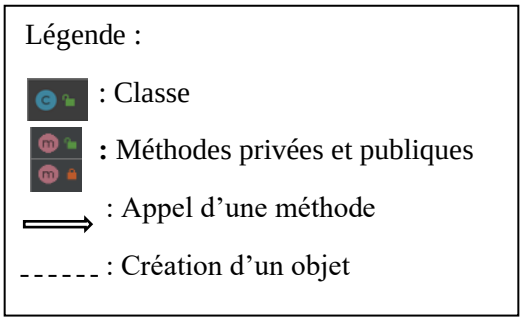


Figure 2 : Diagramme de classe du package game.frogger2.



La classe Main :

La classe *Main* permet de lancer l'application. Elle hérite de la classe Application de JavaFX.

La classe Game :

La classe *Game* représente le jeu. Les méthodes *start ()* et *stop ()* lance et stop le jeu. La méthode *setupGame()* installe les différents éléments du jeu, la méthode *setupScene()* met en place la scène de jeu. La méthode *collisionCheckerMethod()* vérifie s'il y a eu collision et est lancé par un AnimationTimer.

La classe Menu :

La classe *Menu* permet de mettre en forme l'interface graphique du jeu, notamment le menu de démarrage du jeu. La méthode *buttonActionSetup()* est appelé dans la méthode *start* du Main, ainsi le jeu lance à condition que le joueur clique sur le bouton play.

Package *util*

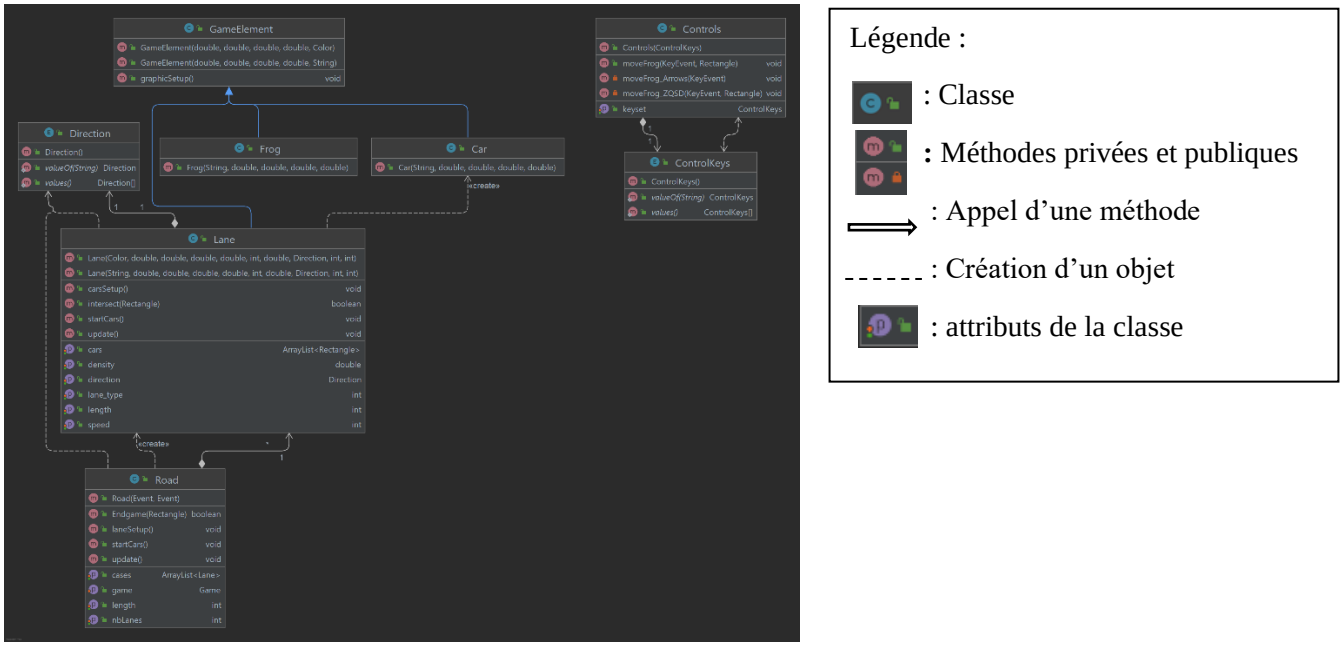


Figure 3 : Diagramme de classe du package util.

La classe Controls :

La classe *Controls* permet de contrôler les mouvements de la grenouille avec la méthode *moveFrog()*. Cette méthode fait appel aux deux méthodes *moveFrog_Arrows()* et *moveFrog_QZSD()*, la première permet de diriger la grenouille avec les flèches du clavier, la seconde fait de même avec les touche des lettres Q,Z,S et D.

Pour permettre au joueur de diriger la grenouille avec les touches de son clavier, la propriété *ControlKeys*, créé dans le package ***util***, est utilisée et est appelé *keyset*. Ce *ControlKeys* peut être utilisé via un getter et un setter. Si le *keyset* vaut ZQSD, alors la méthode *moveFrog_ZQSD* est utilisée. Idem pour les flèches. Ensuite, la direction choisie par le joueur (Z, Q, S ou D) est récupérée à l'aide de *keyEvent.getCode()* pour faire bouger la grenouille dans la direction choisie.

La classe GameElement :

La classe *GameElement* est la classe parent des différents éléments du jeu, comme la grenouille, les routes et les voitures. Elle hérite de la classe Rectangle de Java.

La classe Road :

La classe Road correspond à la route du jeu sur laquelle sont créés plusieurs *lanes*.

La classe construit une liste de *lane*. Le premier *lane* de la liste correspond à la ligne de départ de la grenouille, le dernier correspond à la ligne d'arrivée. La construction de ces *lanes* se fait avec la méthode *laneSetup()*.

La méthode *laneSetup()* permet aussi de rendre les voies deux à onze dangereuses, c'est-à-dire que des voitures et des obstacles sont susceptibles de se trouver sur ces voies. Les voies une et douze sont considérées comme sécurisées pour la grenouille puisqu'elles correspondent à la

ligne de départ et d'arrivée. Ainsi, cette méthode donne de façon aléatoire la vitesse et la direction des voitures sur chaque voie dangereuse.

La méthode *Endgame(Rectangle)* vérifie qu'il n'y a pas eu de collision avec la grenouille sur chacune des voies. Elle prend en argument un objet Rectangle appelé frog.

La classe Lane :

Cette classe correspond aux voies du jeu, les lanes créés dans la classe Road. Les méthodes *carsSetup()* et *startCars()* ajoutes les voitures aux voies et les font démarrer de sortes qu'il y ai assez de voitures dans le jeu sans que celles si ne se superposent.

La classe Car :

Cette classe représente les voitures que doit éviter le joueur. Cette classe hérite de GameElement, elle hérite donc aussi de Rectangle.

La classe Frog :

Cette classe correspond à la grenouille, elle hérite aussi de GameElement.

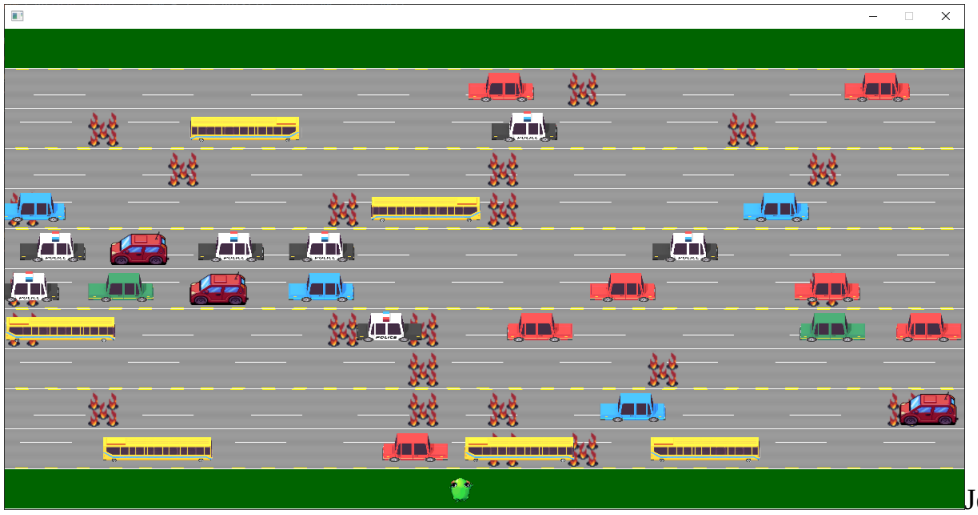
Les classes Direction et ControlKeys :

Ces classes sont des énumérations. Direction énumère les quatre directions up, left, right et down. ControlKeys énumère les deux keyset possible : *arrows*, pour les flèches du clavier, et *ZQSD*.

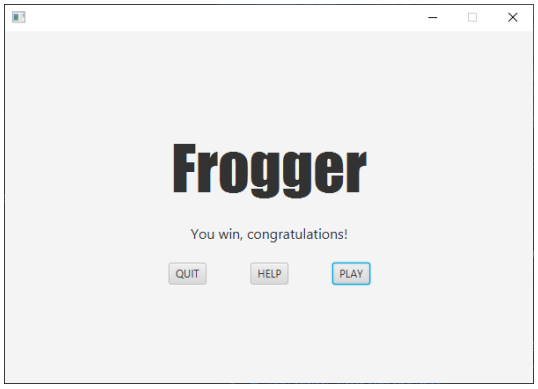
4 – L'amélioration du code :

4.1 – Le jeu :

Menu de départ :


Jeu :

Menu de fin :



4.2 – Un projet agile :

Le projet a été restructuré de manière à séparer la partie application, dans le package ***game.frogger2***, comprenant les animations et le graphisme et la partie « cœur du code »,dans le package ***util***, avec les différents objets manipulés comme la grenouille et les différentes voitures et voies. De cette manière le projet est construit de façon agile : on peut modifier le graphisme du jeu sans modifier le reste du code, le graphisme pourrait aussi être utilisé pour un autre jeu. Aussi, il est possible de modifier la difficulté du jeu, en modifiant la vitesse des voitures par exemple, sans avoir à modifier la partie application.

D'autre part, chaque package est constitué de classe telle qu'un fichier corresponde à une seule classe. De cette façon le code est structuré et permet une lecture claire des différents éléments.

5 – Méthode de travail :

L'utilisation de Github a permis une méthode de travail interactive tout au long du projet. Aussi, le canal teams de l'équipe a été utilisée pour partager les idées et travail en équipe et à distance lorsque la situation l'obligeait.

5.1 – La Répartition du travail

Dans la mesure où chacun des membres du groupes a des domaines de compétences différents, le groupe c'est divisé de la manière suivante afin de permettre à chacun de s'investir dans le projet : Emile Berteloot s'est approprié la partie technique du projet, notamment en développant

une grande partie de l'application. La structuration du projet et du code a été dirigée par Corentin Goetghebeur dans la mesure où cet élément de l'équipe est celui qui avait le plus d'expérience sur le langage de programmation utilisé. Enfin Rozenn Jézégou s'est appropriée la gestion du projet d'une façon transverse. Cependant, cette répartition des tâches donne une tendance sur ce qui a été fait par les membres du groupe. Chacun des éléments s'est intéressé aux tâches attribuées aux autres.

5.2 – Organisation du travail

SEPTEMBRE																	
13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30
Structuration du projet, brainstorming avec toute l'équipe									Ecriture du code, création de l'interface et de la grenouille								

OCTOBRE																	
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
Ecriture du code, création de l'environnement : routes, lanes et voitures									Ecriture du code, création des transitions et des règles du jeu								
									Rédaction du rapport, réorganisation du travail au sein de l'équipe.								

19	20	21	22	23	24	25	26	27	28	29	30	31	
Ecriture du code, restructuration du code, amélioration des règles de jeu									Ecriture du code, amélioration de l'interface graphique				
Rédaction du rapport, préparation de la présentation.									Rédaction du rapport				

NOVEMBRE				
1	2	3	4	5
Amélioration du code et des règles de jeu			Rendu	Présentation
Préparation de la présentation				

Ci-dessus, le calendrier que nous nous sommes fixés pour réaliser ce projet.

6 - Conclusion

Le projet Frogger a permis aux membres de l'équipe de gagner en compétences. D'un point de vue technique, l'équipe a pu prendre en main l'outil Java et JavaFX et élargir ses compétences dans le domaine de la programmation. Aussi, l'équipe s'est confrontée aux enjeux du métier de l'ingénieur : délivrer un produit dans le délai imposé en respectant au mieux le cahier des charges imposé. D'un point de vue social, réaliser un projet en équipe représente toujours un challenge et permet aux différents membres d'apprendre à travailler avec les autres.