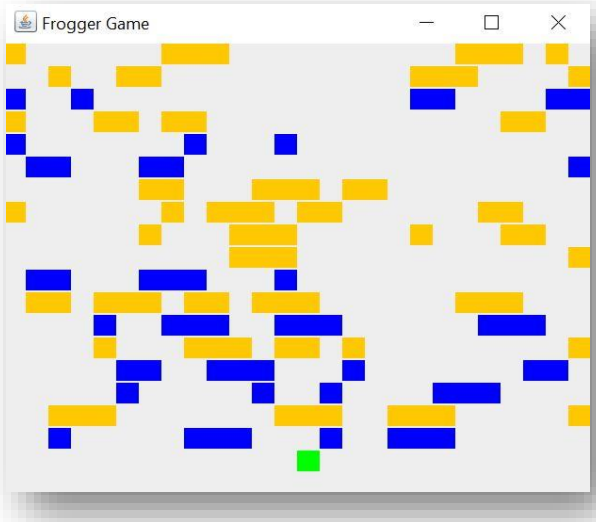
PROJET FROGGER

LOGVINOVA IULIIA
NGUYEN THI THUY TRANG

GROUPE 4 L2 INFORMATIQUE |2019

Introduction :

Notre but était de coder un jeu basique qui consiste à faire traverser une rue à une grenouille en évitant les voitures qui y circulent. Donc, d'amener une "grenouille" du bas de la fenêtre de jeu au haut de celle-ci. Le projet est composé de plusieurs parties.



Les règles du jeu :

La grenouille commence par la 1ere ligne et gagne si elle arrive à la dernière ligne sans accident *(sans impactes avec les voitures).* Les voitures sont générées aléatoirement de deux côtes. Les voitures d'une même voie se déplacent à la même vitesse et dans la même direction *(de gauche à droite ou de droite à gauche).* Les voitures sont de taille 1 à 3.

1ere partie : La grenouille

Pour la première partie il faut compléter la classe ***Frog*** de package ***frog*** qui décrit la grenouille. Ainsi apporté quelques modifications dans la classe ***Game*** du package ***gameCommons***. Dans le package ***gameCommons*** nous avons l'interface ***IFrog***. Nous implémentons et complétons les méthodes qui étaient dans l'interface ***IFrog***. Donc, les getters et une nouvelle méthode *move()* qui définit les mouvements de grenouille. Pour mieux visualiser les mouvements, on affiche à chaque déplacement la direction (*Key pressed up | down | right | left*). Dans le ***Game***, on ajoute deux méthodes : *testWin()* et *testLose()*, elles testent si la grenouille est à position gagnante ou pas. Nous utilisions les méthodes décrites dans ***IEnvironment*** : *isWinningPosition()* et *isSafe()*.

C'était la totalité des taches qu'on devait rajouter. La première partie était relativement simple.

2eme partie : L'environnement

Le but de cette partie était d'implémenter notre propre environnement : les voitures et les lanes (voies). La deuxième partie est dans son propre package ***environement***. Pour chaque partie nous créons un nouveau package sans modifier le reste du projet. Dans ce dernier package nous trouverons 3 classes : ***Car***, ***Lane***, ***Environment***.

Dans la classe ***Main*** du package ***gameCommons*** il faut commencer par remplacer une ligne du code. Avant nous avons utilisés le jar ***givenEnvironment***, mais maintenant nous créons notre propre environnement. Donc, nous avons plus besoin de cette liaison avec le jar fourni, mais une liaison avec notre classe ***Environment*** :`IEnvironment env = new Environment(game)`. Dans le package ***gameCommons*** nous disposons de l'interface ***IEnvironment***. Cela exige d'implémenter les méthodes définies dans ***IEnvironment*** dans notre classe ***Environment*** du package ***environement***. ***Environment*** a deux attributs : lanes et game. Game est notre fenêtre de jeu. Lanes ou voies sont des ArrayLists qui contiennent les voitures. Dans la classe ***Car*** nous créons la méthode *reachesEdge()* qui vérifie si nous dépassons pas la frontière, *goMove()* qui fait bouger la voiture, *caseCovered()* qui vérifie si il n'y a de collision de grenouille et de la voiture, *addToGraphics()* déjà fourni permettant d'ajouter un élément graphique correspondant à la voiture. Pour la classe ***Lane*** nous avons méthodes : *update()*, *hideCars()*, *removeCars()*, *moveCars()*, *isSafe()*.

Dans la 2eme partie c'était nous qui devait coder presque tout, donc c'était un peu plus difficile que la 1ere partie.

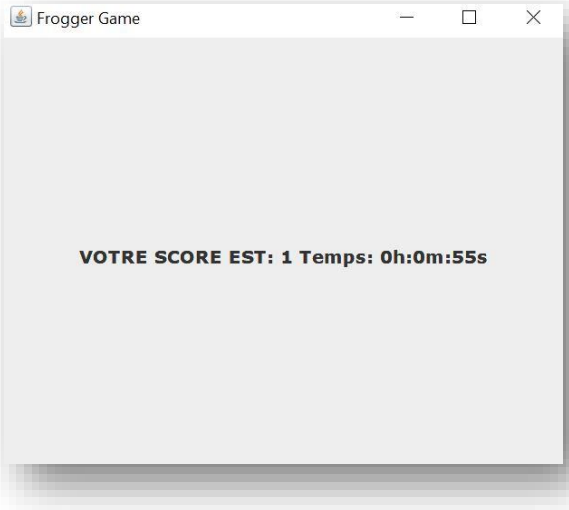
3eme partie : Jeu infini

Dans cette partie c'est le même monde, mais il est infini : il n'y a maintenant plus de ligne d'arrivée. Donc, à chaque déplacement vers haut, nous allons ajouter une ligne de plus en haut et nous allons cacher la ligne en bas (mais pas supprimer, juste cacher). Nous créons un nouveau package ***infiniteGame*** et nous recopions les classes de package ***environement***, nous allons les modifier. Alors, nous aurions les classes ***InfiniteCar***, ***InfiniteEnv***, ***InfiniteFrof*** et ***InfiniteLane***. Dans le ***InfiniteFrog*** toutes les méthodes sont pareilles que dans le ***Frog***, sauf que nous ajoutions `Game game = this.game;` dans chaque condition(up, down, etc.). Pour rendre le jeu infini, nous avons besoin de deux fenêtres, deux games. A chaque déplacement vers le haut, on va déplacer un des games. Comme ça on peut cacher les lignes en bas. Jouer jusqu'à infinie, en s'arrêtant jamais, n'a pas trop d'intérêt, donc dans notre version, on arrête le jeu dès qu'on atteint le score maximal. ***InfiniteCar*** a les mêmes méthodes que ***Car***. ***InfiniteEnv*** répète l'***Environment***, sauf chaque fois pour cacher la ligne en bas, il faut partir de deux et pas d'un : `for (int i = 2; i < game.height; ++i)`. ***InfiniteLane*** est presque la même que ***Lane***, juste quand on rajoute une ligne en haut à chaque fois, il faut aussi rajouter des voitures sur cette nouvelle lane.

La 3eme partie n'était pas tellement difficile et n'a pas posé trop des difficultés, comme essentiellement c'était les mêmes classes que dans le package ***environment***. Il fallait juste les modifier un peu.

4eme partie : Éléments complémentaires

Concernant la 4eme partie nous avons implémenter un timer permettant de compter le temps passer sur une partie de jeu. Tout d'abord nous avons commencer par rajouter une variable timer de type entier dans la classe ***Game***, celle-ci sera incrémenter lorsque le jeu est lancé jusqu'à ce que le joueur gagne ou perd, cependant une autre classe dans le package ***util*** a été créé dont le nom est ***SecToHours*** ayant une méthode start permettant de convertit un entier en série temporelle 'HH :mm : s'.



Conclusion :

Alors, la partie la plus difficile pour nous c'était la partie deux plutôt. Nous avons eu beaucoup de problèmes avec cette partie. A un moment nous avons eu les voitures d'une seule couleur et de la même direction. Dans 3eme partie nous avons eu un bug avec les voitures : même s'il y avait une collision avec une voiture, mais la programme nous laissait continuer. Même maintenant ça prend beaucoup du temps pour arrêter le jeu. Nous avons aussi une petite problème d'affichage de score, même quand on arrête le jeu parce que on a perdu, on peut encore bouger sans voir l'écran. Nous s'intéresser vraiment à la cinquième partie avec les décorations et interface graphique, mais nous n'avons pas eu le temps pour traiter cette partie. En général, c'était un intéressant projet qui nous permit d'appliquer ce que nous avons appris pendant les cours de Java.