Haute Ecole **Libre** de Bruxelles

2021-2022

Programmation III : JAVA

Rapport de projet : Frogger

Jessy ADAM

Professeur :

M. Riggio

2021

Table des matières

1 Introduction

Dans le cadre du cours de Programmation III : JAVA, nous avons été amenés à faire comme projet un jeu basé sur le jeu d'arcade Frogger de 1981. Celui-ci aura été fait sur une base fonctionnelle consistant d'un jeu Snake, fournit au cours.

Certaines contraintes ont du être respectées aussi bien au niveau des fonctionnalités que les librairies utilisées, en plus de devoir faire des rajouts originaux.

Ce rapport concerne donc ma version de ce projet. Celui-ci traitera des fonctionnalités implémentées accompagné d'une analyse de celui-ci.

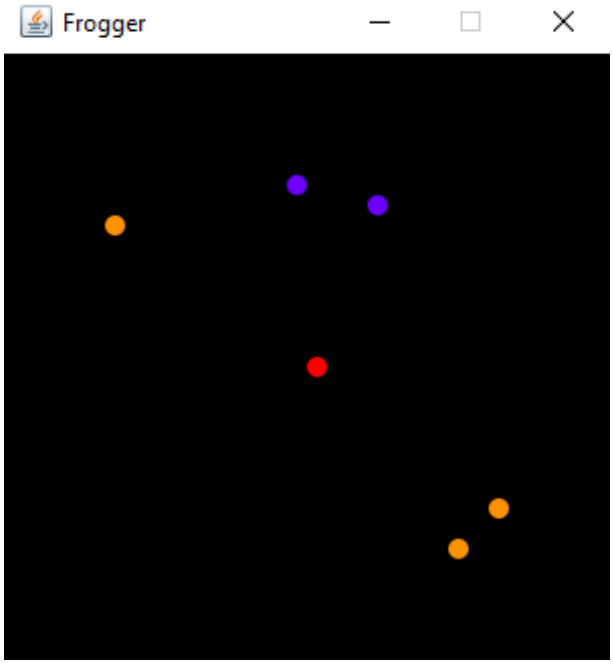
FIGURE 1.1: Aperçu du projet final

2 Fonctionnalités implémentées

Pour ce qui est des fonctionnalités, certaines étaient obligatoires, tant dis que d'autres étaient de mon propre accords, je les ai donc séparés en différentes catégories, en plus d'une brève description de la base fonctionnelle.

2.1 Fonctionnalités de base

La base fonctionnelle fournie au début du projet contenait déjà certaines fonctionnalités.



Elle contenait déjà un système de pièces et d'insectes à collecter, et un système de mouvement, gentiment mit dans une zone de jeu au fond noir. Offrant une bonne base pour commencer.

2.2 Fonctionnalités obligatoires

Ensuite, il était obligatoire d'ajouter des fonctionnalités comme indiqué dans les consigne de projet.

Mais avant de commencer, j'aimerais préciser que les niveaux ne sont pas générés aléatoirement, ils sont à la place parcourus dans l'ordre à partir d'un tableau d'objets "Level", qui ressemblent à ceci :

```
new Level("Route de campagne", "GGGWWWWWWWGGRRRRGGWWWWWW", new int[]{ 40, 0, 20, 20, 20 }, 3, 1, 1, 0.1, 0.25, BACKCOLOR),
new Level("Au milieu d'un champ de blé", "GGGRRRGGWWGGRRRRGGGRRR", new int[]{ 60, 10, 10, 10, 10 }, 4, 1, 2, 0.25, 0.25, Color.decode("#E5CA54")),
```

Ceux-ci sont sollicités par la suite notamment à l'initialisation d'une partie.

2.2.1 Terrain de jeu

Le terrain de jeu est géré par une chaîne de caractère à indiquer pour chaque niveau, allant de haut en bas, le reste étant complété par de l'herbe.

```
", "GGGRRRGGWWGGRRRRGGGRRR",
"GGGRRRRGGGGRRRRGGGGRRRR",
```

FIGURE 2.1: Exemple de chaîne de caractère de terrain

Pour l'instant, il y a trois types de terrains disponible pour chaque niveau :

— L'**herbe**, qui prend une couleur au choix et ne change pas grand chose à part pour des buissons qui peuvent y apparaître, si le niveau le veut.

— La **route**, où un certain nombre de voitures sont disposés sur chaque bande au début de la partie et sur lesquelles elles se tiendront.

— Et la **rivière**, où sont placés des tronc d'arbres de manière pseudo-aléatoire (pour éviter de rendre le niveau impossible) sur lesquels Frogger peut se déplacer. Et où s'il tombe dans l'eau, il perd une vie. A noter que même si les tronc sont placés de manière à être obligatoirement possible à traverser, il peut y avoir des tronc isolés, ce qui fait que le niveau peut être impossible si une pièce est placée dessus.

FIGURE 2.2: Exemple des trois types de terrains

2.2.2 Collectibles et autres éléments du jeu

Pièces

Après la génération du terrain partir des données du niveau, on y ajoute ensuite un nombre prédéterminé de pièces à collecter sur toute surface où Frogger peut aller (ce qui se résume à pas sur l'eau, ni sur un buisson, ni sur le goal en haut de l'écran, par souci esthétique).

Insectes

Des insectes sont ensuite disposé de la même manière sur le terrain, bien qu'il y en ait plusieurs types, le nombre est calculé de manière a ce que pour chaque X nombre de pièce, un insecte est ajouté.

FIGURE 2.3: Aperçu des types d'insectes

Pour l'instant il y en a trois :

— **Une abeille(?) rose**, comme insecte commun, apparaissant à chaque pièces et donnant 2 points.

— **Un scarabée magenta**, relativement rare, apparaissant chaque 3 pièces et donnant 3 points.

— **Une coccinelle mauve**, très rare, apparaissant toutes les 5 pièces et donnant 5 points.

Buissons

Sur les bandes d'herbes sur lesquelles Frogger n'apparaît pas au début du niveau, on dispose un nombre déterminé par bande de buissons que Frogger ne peut pas franchir, comme obstacles. Ces buissons ont trois types pour de la variété visuelle : un sans rien, un avec une fleur et un avec des fruits quelconques.

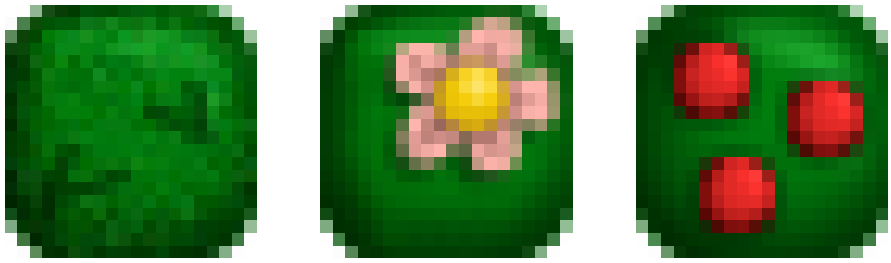


FIGURE 2.4: Aperçu des types de buissons

Á noter que rien n'empêche aux buissons d'être aléatoirement placés au même endroit, ce qui fait que certaines bandes pourraient manquer d'un buisson où deux. Leur collision est aussi basée sur la direction qu'à Frogger ce qui pourrait poser problème, s'il arrive a changer de sens pendant la vérification de celle-ci.

Voitures

Après cela des voitures sont placées par groupes, dont le nombre est déterminé par l'information du niveau, sur chaque bande avec une direction et position aléatoire. La vitesse est aussi aléatoire mais est contrainte à une certaine intervalle choisie par le magnifique game designer que je suis.

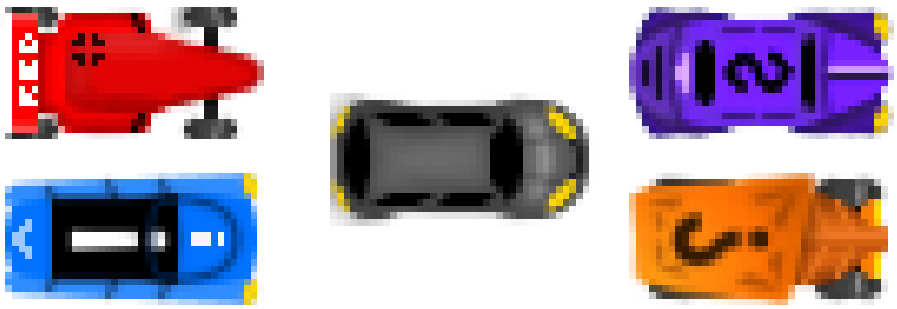


FIGURE 2.5: Aperçu des types de voitures

Il existe 5 types de voitures :

— **Blinky**, une voiture rouge, qui suit Frogger dans les limites de sa route. Un petit problème avec est que permettre une vitesse de plus de deux bandes le permet de sortir de la rue et d'aller chez Frogger sans qu'il n'aie besoin de bouger, due à la manière dont la collision avec la route est calculée.

— **Pinky**, une voiture ironiquement violette, qui accélère a chaque fois qu'on prend une pièce. A noter que pour rendre l'effet plus prononcé, sa vitesse n'est pas aléatoire mais est toujours la vitesse minimale du niveau au début, et la vitesse maximale avec toutes les pièces collectées.

— **Inky**, une voiture bleue, qui ralenti si Frogger est sur la même bande que lui, même qui ce n'est qu'une partie de Frogger.

— **Clyde**, une voiture orange, qui change de vitesse aléatoirement (vitesse restant quand même dans les limites imposées du niveau) et qui peut aussi rarement changer de sens de manière aléatoire, pour rajouter un élément de surprise.

— et finalement, une voiture que j'aurais probablement pu appeler **Orson**, mais qui est juste appelée **Voiture**, grise et tout a fait normale, allant de gauche à droite, ou inversement. Bien qu'elle n'aie pas été mentionné dans les consignes, j'ai décidé de l'inclure pour éviter d'avoir *trop* de chaos.

Le type d'une voiture est sélectionné de manière aléatoire entre les types disponible en fonction d'un système de probabilité. En indiquant un tableau de valeurs représentant la probabilité de chaque voiture dans les informations de niveau, la fonction responsable de piocher une voiture le fera en fonction, ce qui fait qu'on peut favoriser ou exclure telle ou telle voiture.

Pilule

Finalement une pilule est disposé quelque part sur le terrain, une fois collectée, elle rend Frogger bleu, invincible et lent, pendant 10 secondes, permettant à Frogger de manger, probablement, les voitures qui passent sur son chemin donnant 2 points chaque. Pendant qu'il est invincible, tout les points que Frogger gagne pendant ce temps sont doublés.

Le temps restant à Frogger est aussi indiqué en haut de l'écran avec un pilule représentant une seconde.

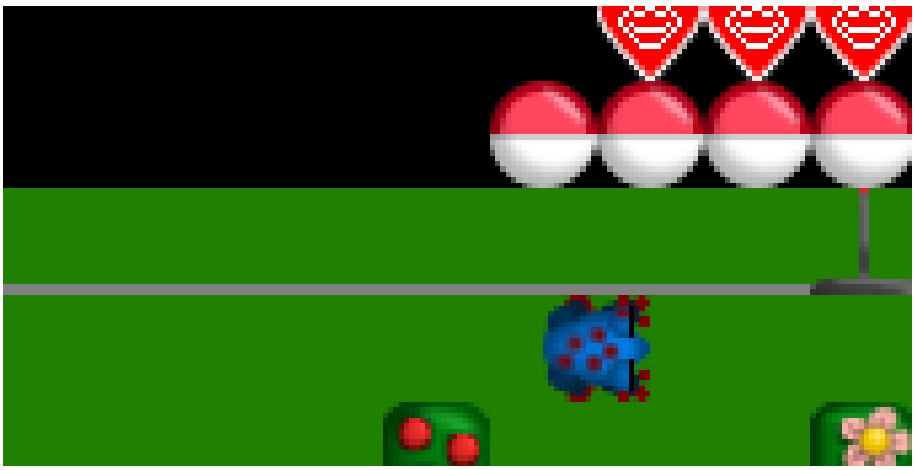


FIGURE 2.6: Aperçu de Frogger en période d'invincibilité

La pilule a aussi une chance sur 20 de devenir Pierre qui roule. Je ne donnerait pas plus de précisions.

Frogger

Quelques notes sur notre protagoniste. Contrairement au consignes, il se déplace par longueur de bande à la fois à l'horizontal et à la verticale par simplicité de compréhension. Une fois invincible, il va 2 fois plus lentement ce qui veut dire qu'il va par demi-cases. S'il n'est pas aligné une fois son invincibilité terminée, le jeu essayera de l'aligner lors de son prochain mouvement dans cette direction. Ce système bien que fonctionnel peut très vite casser si Frogger ne se déplace pas par un multiple d'une case en temps normal et assume que l'invincibilité est la seule chose qui peut désaligner Frogger.

2.2.3 Déroulement de partie

Après une transition indiquant le niveau à suivre, celui-ci est généré.

FIGURE 2.7: Transition aux niveau 5

Quand le joueur a ramassé toutes les pièces la ligne d'arrivée en au de l'écran change d'aspect, au quel moment il doit l'atteindre pour terminer le niveau et passer au suivant. Répéter autant de fois que nécessaire jusqu'à ce qu'on arrive au dernier niveau, à la fin duquel on ajoute un certain nombre de points au score en fonction des vies restantes et on change le meilleur score si besoin. On affiche ensuite cela pour après tout remettre à zéro et recommencer le jeu.

FIGURE 2.8: Aperçu de la ligne d'arrivée avant et après avoir tout ramassé

Si Frogger meurt à un moment dans un niveau, il perd une vie et recommence après une transition et une régénération du niveau. S'il perd toutes ses vies, on l'affiche, on change potentiellement le meilleur score et on recommence tout.

Une bonne partie de cette information est affichée sur le HUD, en haut de l'écran, qui comprend :

— Votre score actuel ;

— Le meilleur score, qui comme un bon NES, ne se sauvegarde pas après fermeture ;

— Le niveau courant, ainsi que le bonus que j'expliquerais plus tard ;

— Le nombre de pièces restantes à collecter dans le niveau ;

— Vos vies restantes, ainsi que le maximum ;

— et, si vous êtes invincible, le temps restant de votre invincibilité.

FIGURE 2.9: Aperçu du HUD

2.3 Fonctionnalités supplémentaires

Au niveau des ajouts personnels, j'ai beaucoup fait de petits ajustement mais il y en a quand même deux assez notable.

2.3.1 Système de niveau

Comme dit plus tôt 2.2, la plupart du jeu repose sur un système de niveau qui dicte un bon nombre de choses, comme :

— Son nom, indiqué sur les transitions ;

— Son terrain ;

— La probabilité d'apparition des types de voitures ;

— Le nombre de pièces ;

— Le nombre de voitures et de buissons par bande ;

— Le vitesse disponibles ;

— La couleur de l'herbe ;

Ce qui fait qu'on peut devenir un minimum créatif avec les niveau, tout en essayant de bien gérer la difficulté et de donner des défis différents. J'ai, par exemple, pu faire une plage et un embouteillage.

FIGURE 2.10: Niveau de la plage

FIGURE 2.11: Niveau de l'embouteillage

Un problème de ce système, est qu'il n'y a pas de limite imposée, on peut par exemple mettre une route là où apparaît Frogger, le mettant immédiatement en danger. On peut mettre assez de voitures ou de buissons que le niveau en devient littéralement impossible. On peut parfaitement coincer une pièce entre 4 buissons magnifiquement bien placés et probablement d'autres auxquels je ne pense pas.

2.3.2 Bonus de temps

J'ai aussi ajouté un bonus de temps, qui diminue chaque seconde, et qui est ajouté au score une fois à l'arrivée, rien ne se passe s'il arrive à 0, juste au cas où il y a un niveau complexe à naviguer.

Comme ça, le joueur est encouragé à équilibrer vitesse et points en allant aussi vite que possible tout en ramassant un maximum d'insectes.

2.3.3 Autres

J'ai aussi fait quelques petites choses en plus, comme :

— L'ajout d'une voiture normale ;

— Des graphismes originaux de ma part ;

— D'autres détails graphiques comme un bord visible sur les rivières et des bandes de séparations sur les routes, généré dynamiquement ;

— Des magnifiques transitions ;

— Les tronc s'enfoncent un peu dans l'eau si Frogger est dessus.

— La pilule peut avoir un aspect différent si on est chanceux.

Bien que ces fonctionnalités supplémentaires ne sont forcément pas très conséquentes, ma priorité était d'essayer de faire un jeu le plus jouable et amusant possible avec ce que je pouvais faire.

3 Analyse

Voici un diagramme montrant la structure des classes de ce jeu, avec les flèches pleines représentant de l'héritage directe et celles en pointillés une implémentation d'interface.

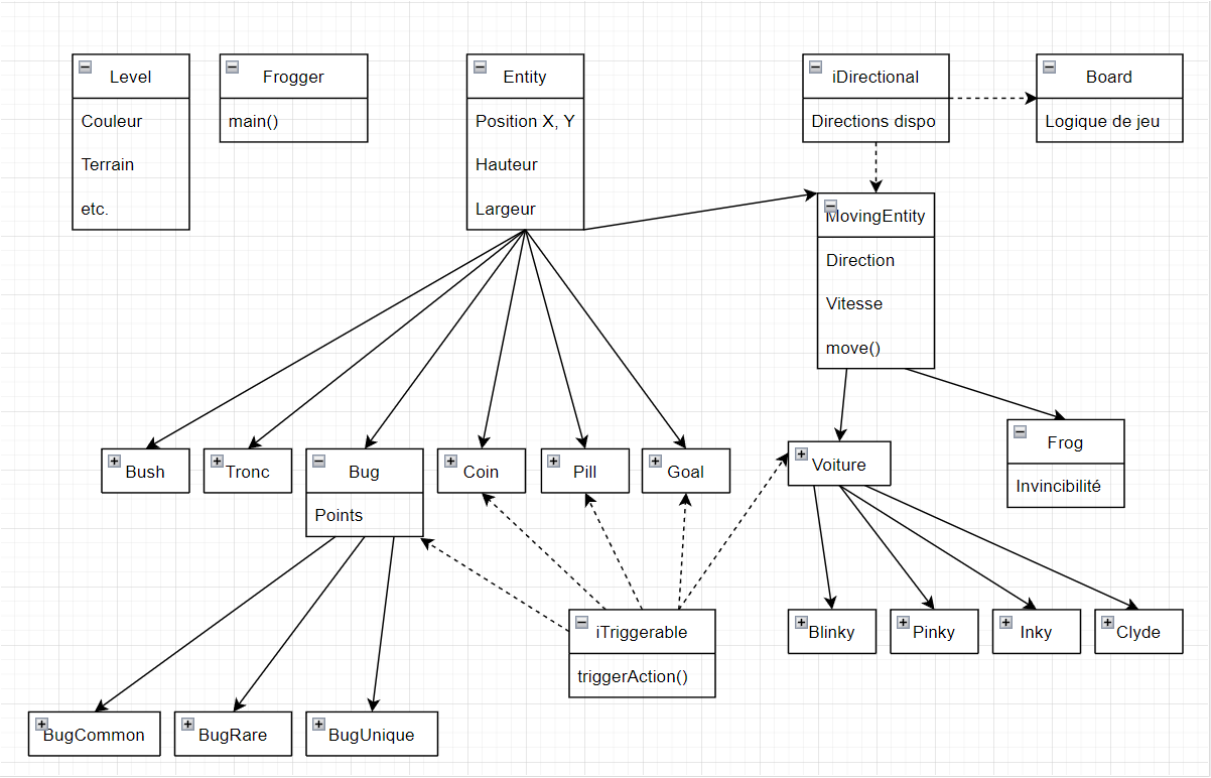


FIGURE 3.1: Structure approximative du jeu

Vous pouvez y voir que la classe Entity est la base de tout ce qui n'est pas la logique du jeu même ou l'information de niveau. Elle est composée d'une position verticale et horizontale, ainsi que d'une hauteur, largeur et de méthodes de collision.

D'elle s'écoule directement toute entité qui ne bouge pas comme la ligne d'arrivée, les troncs, les pièces et les insectes. Les types différents d'insectes héritent donc d'une classe comme par souci de réduction de code.

Tout ce qui peut bouger hérite de MovingEntity, une spécialisation d'Entity qui possède une direction, une vitesse et une méthode de mouvement. La voiture (représantant la voiture normale) est aussi la source des voitures plus spécialisés.

Les directions sont fournies par une interface iDirectional pour éviter les incohérence et afin de pouvoir les changer comme on veut. Tout ce qui peut interagir avec une autre entité ou le jeu implémente aussi iTriggerable afin de pouvoir y faire des changement si nécessaire, tel qu'augmenter le score.

4 Conclusion

Ceci conclut donc ce rapport de ma version du jeu Frogger pour le cours de Programmation III : JAVA, dans celui-ci nous avons parlé des fonctionnalités implémentées en plus d'une courte analyse de sa structure. Je suis assez fier de celui-ci et j'espère qu'il sera à au moins une partie de vos exigences. Sur ce j'ai 20 minutes pour tout relire, donc merci d'avoir lu et potentiellement joué au jeu.