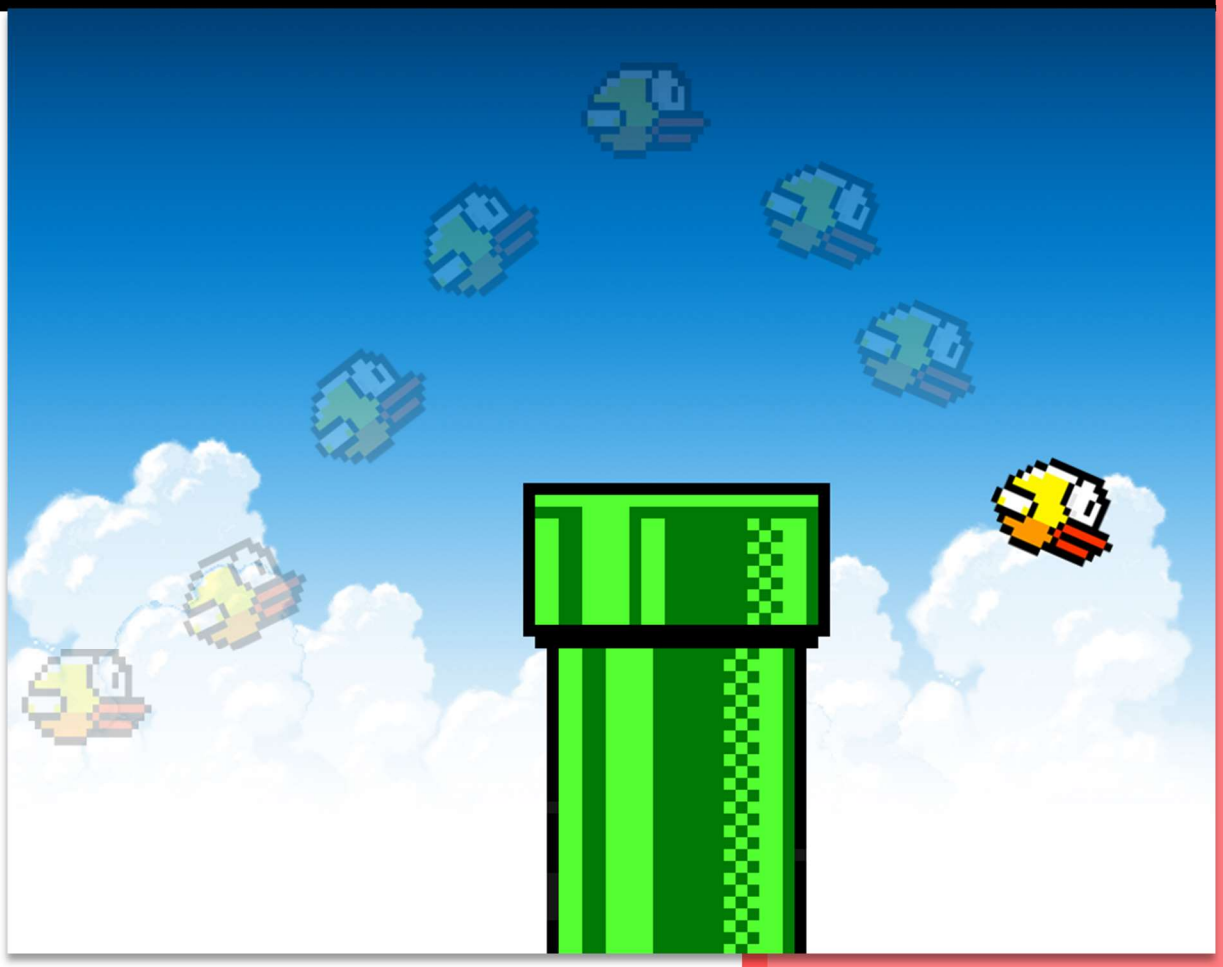
2020

FlappyBird - Rapport de projet



Bovay Louis

DIVTEC

05/11/2020

Table des matières

1. Biographie

Je suis Louis Bovay, apprenti informaticien de 3ème année à l'EMT de Porrentruy.

Cela fait bientôt 3 ans que j'ai commencé à apprendre les bases de la programmation et bientôt 2 ans pour la programmation orienté objet.

J'ai dû réaliser ce projet lors de l'atelier de programmation orientée objet enseigné par Jérôme Conus.

2. But et contexte du projet

Ce projet a pour but de réaliser un petit jeu tout en utilisant la programmation orientée objet.

Il fait le lien entre la théorie acquise en modules et la pratique que l'on pourrait retrouver en entreprise.

2.1 Histoire du jeu

Le jeu est apparu le 24 mai 2013 sur Android et iOS et fût supprimé des différents Appstore le 9 février 2014 par son créateur car il n'arrivait plus à supporter le succès de son jeu (source).

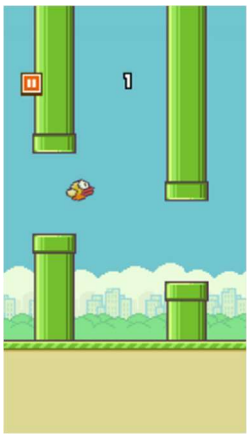


Figure 1 Jeu original - FlappyBird

2.2 Choix du jeu

J'ai choisi FlappyBird car le jeu contient plusieurs fonctionnements intéressants tout en restant extrêmement simple : c'est juste un oiseau qui doit passer entre des tuyaux, mais qui doit pouvoir « voler » fluidement, et les tuyaux doivent être imprévisibles.

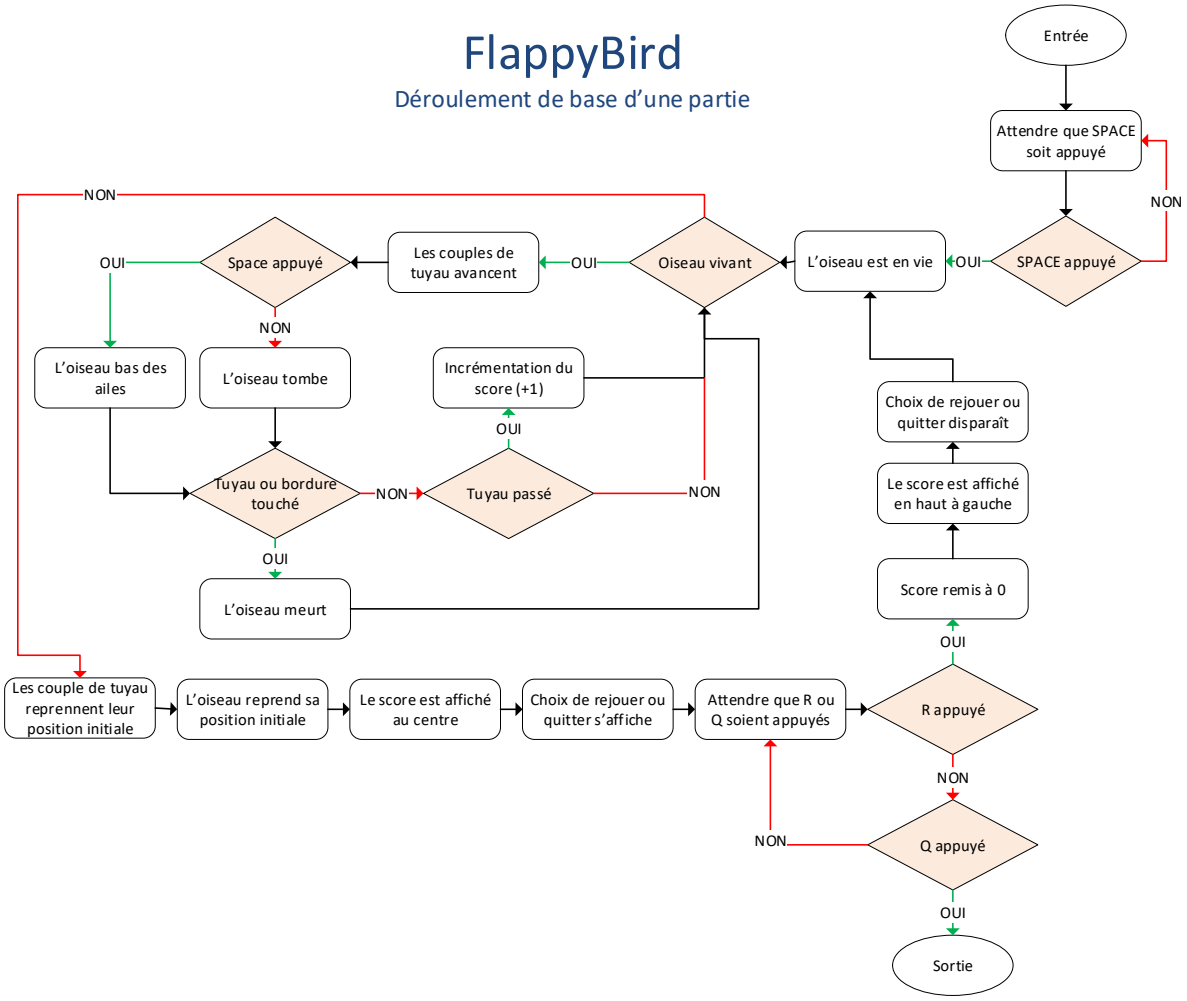
3. Déroulement du projet

3.1 Analyse du jeu

Pour comprendre et analyser le jeu, j'ai utilisé :

- Le jeu (qui est disponible n'importe où gratuitement)
- Mes connaissances en matière de programmation orientée objet

Bien que j'eusse le plan en tête de ce que je souhaitais faire, j'ai créé un ordinogramme du jeu



3.2 Langage de programmation

L'enseignant nous a proposé deux langages : C++ ou Java

Ne connaissant pas le C++ orienté objet j'ai pris la peine d'essayer avant de choisir de partir sur Java/Java FX.

3.3 Les classes principales

Voici les classes principales que j'ai créé afin de représenter les différents objets de mon jeu :

- Bird représente l'oiseau FlappyBird
- Pipe représente un tuyau
- PipeCouple représente un couple de 2 tuyaux
- Shape représente une forme
- Area représente la zone qui délimite une forme

Toutes ces classes et d'autres sont expliquées au sein du projet, leurs spécificités sont détaillées dans la documentation technique.

3.4 Création des sprites

3.4.1 L'oiseau

Image de base récupérée sur internet, j'ai juste raccourci le corp pour qu'il paraisse carré, et fait tourner son aile.

Figure 2 Source : https://www.pngitem.com/middle/iJoJmTi_transparent-background-flappy-bird-hd-png-download/

3.4.2 Tuyaux

J'ai dupliqué le tuyau ainsi que sa base pour le rallongé.

Figure 3 Source : https://www.clipartmax.com/middle/m2i8A0i8N4H7Z5Z5_mario-pipe-pixel-mario-tube-png/

3.4.3 Le fond

J'ai créé un fond blanc d'une fois la hauteur de mon jeu et de deux fois sa largeur, appliqué un dégradé bleu, puis placé des nuages un peu partout, puis j'ai divisé cette image en deux parties pour avoir deux images qui se suivent

Figure 4: https://www.pinterest.com/pin/687010118127637991/

3.5 Réalisation de la documentation

Tout au long du projet j'ai suivi cette façon de faire :

Si je suis bloqué à un moment ou un autre, je fais la documentation (hors Java Doc qui est au contraire directement écrite lors de la programmation) de ce que j'ai déjà réalisé.

Le projet est assez cours et à moins d'être devant un mur infranchissable cette façon de faire m'a semblé être adaptée.

3.6 GitHub

Sur le GitHub 2020-JCO-FLAPPY-BIRD se trouve le projet ainsi que tout ce qu'il faut pour pouvoir l'ouvrir, un build[1], une présentation rapide et toute la documentation.

4. Problèmes rencontrés

4.1 La vitesse du jeu

La fonction update est appeler à chaque tick[2] de processeur, donc si le processeur est puissant, le jeu va vite, s'il ne l'est pas, le jeu sera lent.

```
Il a fallu ajouter un timer qui lance la fonction update chaque
seconde, puis augmenter sa fréquence de 60.
```

4.2 Les sauts

Les sauts ne sont pas fluides : l'oiseau ne fait que voler très vite un peu plus haut et redescendre juste après, ce qui crée un coupure nette entre voler/tomber

```
Au lieu de faire bouger l'oiseau d'un point A à un point B, on lui
donne une vitesse vers le haut plus grande que la gravité, puis on
la réduit progressivement à chaque frame[3], jusqu'à ce que la gravité
reprenne le dessus.
```

5. Stade actuel du projet

5.1 Problèmes connus

5.1.1 Création d'un éxécutable

Après avoir créé un nouveau projet, exporter mon projet sur un autre poste et en suivant beau nombre de tutoriel différent, la création d'un exécutable semble impossible, le problème semble venir du code.

Solution intermédiaire : jouer au jeu depuis l'IDE.

5.2 Fonctionnalités supplémentaires

5.2.1 Animation plus détaillée

L'animation actuelle est simplement : l'oiseau tombe donc ces ailes sont dirigé vers le haut OU l'oiseau vole et ses ailes sont dirigé vers le bas.

Il faudrait donc mettre en place une animation de sprite[4] pour rendre le tout plus esthétique.

```
Au lieu d'avoir mis une animation de sprite, on penche l'oiseau vers
l'arrière quand il bas des ailes, et on le fait piquer du bec quand
il tombe, en plus de ça, l'animation de base se voit et rend mieux.
```

5.2.2 Mode difficile

Un mode difficile où les tuyaux bougeraient en vertical.

```
Le mode difficile est activable directement depuis le menu en
appuyant du G, le tuyaux bougeront alors en vertical tout le long de
la partie. En appuyant sur G à nouveau, le mode normal revient.
```

5.2.3 Tableau de score

Un tableau de score permettant de garder en mémoire un certain nombre de scores.

```
Un tableau des score est maintenant présent pour les deux mode, il
suffit de maintenir TAB sur le menu et le tableau des score
correspondant à la difficulté s'affichera.
```

5.2.4 Timer avant de recommencer une partie

Pour éviter de relancer une partie sans faire exprès, ajouter un délais avant de relancer la partie.

```
Une timer se lance en fin de partie et affiche à la place de la
touche à appuyer pour rejouer un compte à rebours.
```

6. Connaissances acquises

Tout au long du projet j'ai appris différentes choses telles que :

- Si tu as une idée de classe, quelqu'un la surement déjà faite et mise à disposition.
- La mise en place d'une scène ainsi que tout ce qu'elle contient
- Les bases de Java FX
- Mettre enfin en pratique les connaissances acquises en module

7. Conclusion

Il est très satisfaisant de voir graphiquement ce que donne toutes les lignes que nous avons écrites et donne envie d'aller plus loin et de continuer vers cette voie.

Ce projet regroupe une grande partie de ce que j'aime : des casse-tête, l'utilisation de son imagination et les jeux-vidéo.

8. Sources

GitHub du projet : https://github.com/divtec-cejef/2020-JCO-FLAPPY-BIRD

Documentation Technique : https://github.com/divtec-cejef/2020-JCO-FLAPPY-BIRD/blob/main/docs/FlappyBird_DocumentationTechnique.pdf

9. Index

[1] Projet déjà « construit » qui donne accès à un exécutable ainsi de tout ce dont il a besoin pour se lancer
[2] Unité arbitraire de mesure de temps d'un processeur
[3] « Image », à chaque fois que l'écran se rafraîchit, on passe à la frame suivante
[4] Elément graphique d'un objet