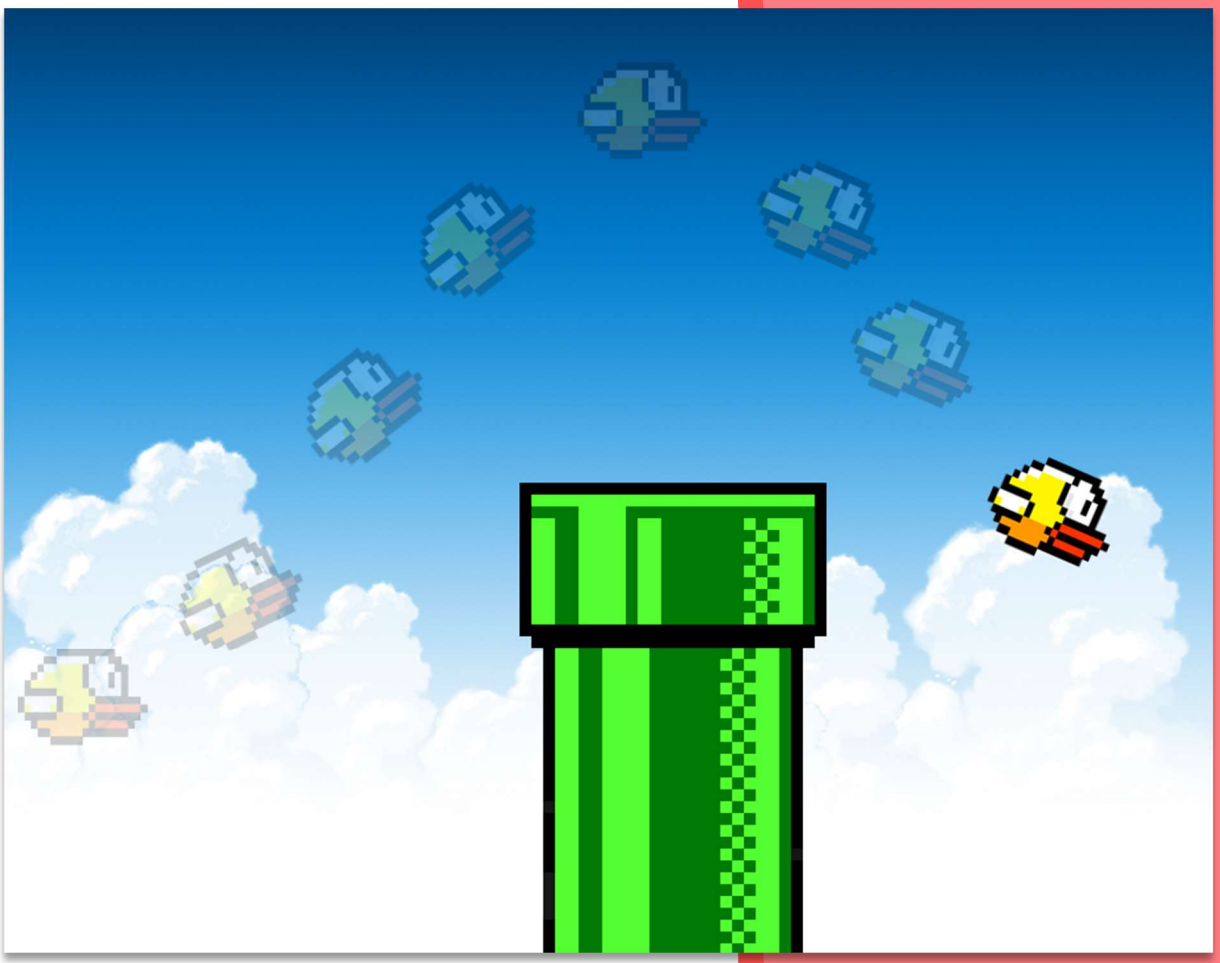
2020

FlappyBird – Documentation technique



Bovay Louis

DIVTEC

05/11/2020

Table des matières

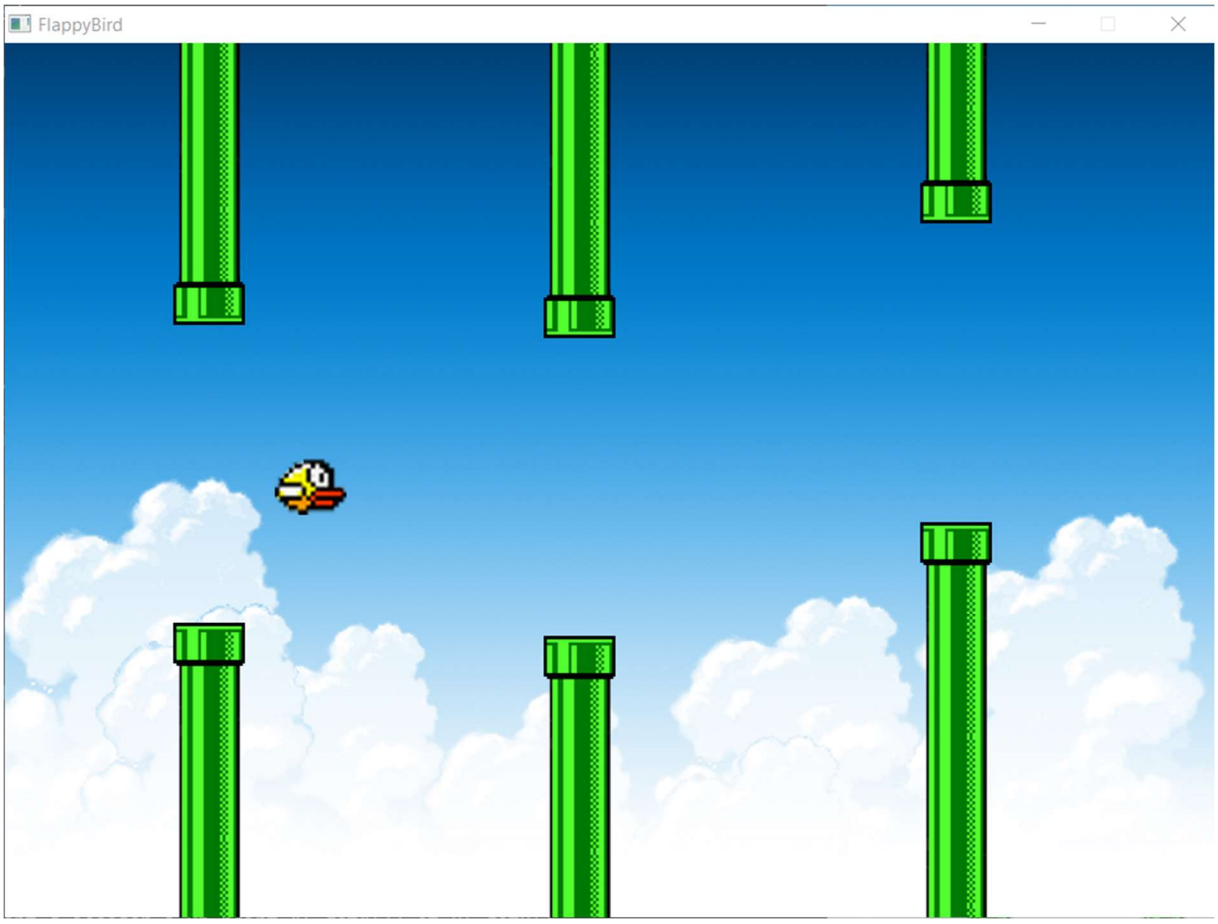
1. Description du projet

Le projet à pour but de réaliser le jeu FlappyBird en programmation orientée objet.

Le but du jeu est de faire battre des ailes un oiseau pour qu'il ne se cogne pas à des tuyaux placés en haut et en bas qui foncent sur lui.

S'il l'oiseau touche un tuyau, le plafond ou le sol, il meurt.

A chaque fois qu'il esquive un couple de tuyaux, un point est marqué.



2. Convention de nommage

- Tous les noms de variable sont écrits en CamelCase
- Tous les noms de fonction doivent décrire l'action faite dans la fonction et sont écrits en CamelCase.
- Tous les noms de classe sont écrits en CamelCase et commence par une majuscule

3. Installations

- IntelliJ: https://www.jetbrains.com/fr-fr/idea/
- Le SDK JavaFX: https://gluonhq.com/products/javafx/
- Tutoriel d'installation de JavaFX par Jetbrain : https://www.jetbrains.com/help/idea/javafx.html#create-project

Créez un nouveau projet et sélectionnez « JavaFX » dans la liste de guahce.

Cliquez sur le menu déroulant « Project SDK » et ajoutez le SDK téléchargé via le bouton « Add JDK… »

Choisissez un nom de projet ainsi que ça location et voilà : le projet inclura les librairies javafx.

4. Les classes :

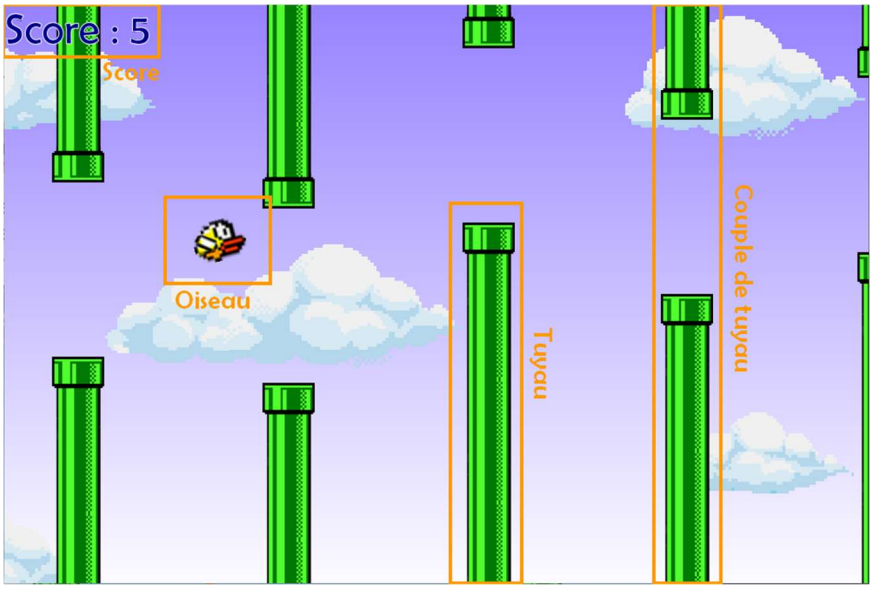
Shape, qui hérite de Rectangle représente un rectangle physique en 2 dimensions qui peut se déplace en XY et donner les coordonnées de ses 4 coins grâce à sa variable Area qui permet de générer les coordonnées des quatre coins du rectangle.

Bird, qui hérite de Shape représente un oiseau qui peut voler, tomber, mourir et revivre.

Pipe, qui hérite aussi de Shape, représente un tuyau qui peut détecter si l'oiseau le touche.

PipeCouple représente un couple de tuyau qui peut bouger vers la gauche, revenir instantanément à droite, générer un espace fixe à un endroit aléatoire sur l'axe Y entre ses deux tuyaux et donner un points au joueur.

Score représente un score, il peut stocker les points, les fournir, les remettre à zéro ainsi que les afficher.



5. Explications supplémentaires

Cette partie est là pour expliquer ce qui ne pourrait pas être facile à comprendre ou pour faciliter l'apprentissage de méthodes utilisées au sein du projet.

Si une fonction / variable est mentionnée ou alors qu'une partie de code est montrée, il sera précisé à chaque début de sujet dans quel `fichier` la trouver.

S'il y a plusieurs fichiers mentionnées, les variables/fonction seront précédées de leur classe.

5.1 Les « fausses scènes »

`Main.java`

En théorie, on pourrait dire que le jeu à trois scènes :

- Le menu principal
- Le jeu
- Fin de partie

En pratique, c'est toujours la même scène qui se joue mais qui change d'état.

Il y a pour ça deux booléen :

```
boolean isGameRunning = false;
boolean isGameStarted = false;
```

`IsGameStarted` sert juste une seule fois pour faire apparaître le menu principal, nous ne sommes pas sensé pouvoir retourner au menu principal, à moins de relancer l'application

`IsGameRunning` sert à savoir si le jeu est en cours ou non, si `isGameRunning` = `true` alors le jeu se joue, si `isGameRunning` = `false` l'écran de score s'affiche et propose de rejouer.

Tout ça sert savoir quand lancer une de ces trois méthodes :

```
public void startGame() {…}
public void restartGame() {…}
public void endGame() {…}
```

Ces trois méthodes servent comme leur nom l'indique à démarrer, redémarrer et finir la partie.

5.2 Les couples de tuyaux

`Pipe.java`
`PipeCouple.java`

Dans le jeu, il y a des couples de tuyaux « `PipeCouple` » qui contiennent deux tuyaux « `Pipe` » chacun et qui bougent de droite à gauche (`PipeCouple.move()`).

Les deux tuyaux présents dans un couple ont un espace fixe de 250, l'espace peut apparaître n'importe où sur leur axe Y (`PipeCouple.createSpace()`).

Arrivé à gauche, ils sont reformatés (`PipeCouple.formatCouples()`) et retrouvent leur état d'origine (à droite, avec un nouvelle espace entre eux et le droit de donner un point) :

```
public void formatCouples() {
    //tuyau du haut
    this.pipe1.setTranslateX(550);
    this.pipe1.setTranslateY(-350);
    //tuyau du bas
    this.pipe2.setTranslateX(550);
    this.pipe2.setTranslateY(350);
    //Ils peuvent à nouveau donner des points
    this.canGivePts = true;
    // Générer l'espace aléatoire entre les deux tuyaux
    createSpace();
}
```

À chaque fois qu'un couple passe la position Y de l'oiseau, leur état passe de « apte à donner un point » à « non apte à donner un point » (`canGivePts`).

5.3 Le background

`Main.java`

Le background du jeu est composé de deux images, la particularité de ces images c'est qu'elles se suivent parfaitement, il n'y a pas de « coupure » entre elles.

Pour l'animation, on les place côte à côte et on les fait se déplacer de droite à gauche.

Comme les tuyaux, lorsqu'un des deux image se retrouve trop à gauche, on la téléporte tout à droite, puis on la refait passer : le background infini est créé.

```
    if (background.getTranslateX() <= -1000) {
        background.setTranslateX(1000);
```

5.4 La classe Area

```
Area.java
Shape.java
Pipe.java
```

La classe Area représente les 4 coins d'une forme de la classe `Shape` qui permettent de former un rectangle.

Un objet Area sera composé de 4 coordonnées XY de la classe CoordXY qui représenteront le coin haut-gauche, haut-droit, bas-gauche et bas-droit.

L'utilité première de cette classe est de détecter les collisions (`Pipe.isHit(Bird bird)`).

Si un des 4 coins de l'oiseau se trouve entre deux points horizontaux et deux points verticaux, cela signifie que l'oiseau a touché une tuyau

Figure 1 Fonctionnement de base de Area

5.5 Gestion du temps et des frames

`Main.java`

5.5.1 Création

Lors de la création de la scène (`createContent()`), on lui assigne une timeline.

Chaque seconde, elle fait quelque chose, ici `update()`), on règle son cycle comme `Animation.INDEFINITE` pour la faire tourner à vie, à moins de l'arrêter manuellement, on la lance et on règle sa vitesse à 60, comme ça on arrive à 60 `update()` par seconde, donc 60 images par secondes.

```
//Timer basé sur 1 seconde
Timeline timeline = new Timeline(new KeyFrame(Duration.seconds(1),
event -> {
    update();
}));
//nombre de fois qu'elle se répète
timeline.setCycleCount(Animation.INDEFINITE);
//Lancer la timeline
timeline.play();
// vitesse 60 = 60 images par seconde
timeline.setRate(60);
```

5.5.2 Gestion

Deux variables sont à disposition au besoin

```
//Indice de temps, 1 = 60 frame
float t = 0;
//Indice de frame, 60 = 1 seconde
int frame = 0;
```

Elles s'actualisent à chaque `update()` et donc représentent chacune 1 seconde sous différentes formes, si par exemple nous souhaitons faire quelque chose toutes les 10 frames, ou toutes les 5 secondes.

```
//Incrémentation des indice de frame et de temps
//Quand t atteint 1, une seconde sera écoulée
//Quand frame atteint 60, une seconde sera écoulée
t += 0.016;
frame++;
```

5.6 Les sauts fluides

```
Bird.java
Main.java
```

Trois paramètres entre en compte concernant les sauts fluides :

- La gravité (`Main.BIRD_GARVITY`)
- La vitesse de pointe (`Bird.momentum`)
- La perte de vitesse (gérée en modifiant `momentum` dans `Bird.smoothFlap`)

Le la vitesse de pointe doit toujours être de base supérieure à la gravité.

Au fil du temps, la perte de vitesse s'accumule jusqu'à ce que la gravité reprenne le dessus.

```
public void smoothFlap() {
    if (momentum > 0) {
        this.moveUp((int) momentum);
        momentum -= 0.5;
        birdSprite.setRotate(-20);
    } else {
        flying = false;
    }
}
```

`birdSprite.setRotate(-20)` sert à faire se pencher l'oiseau vers l'arrière pour donner l'impression qu'il tire vers le haut pour voler.

Il faut donc faire variés ces trois paramètre pour obtenir un saut fluides, pas trop arqué et qui ne donne pas l'impression d'être sur la lune.

Paramètres de base :

```
Main.BIRD_GRAVITY = 8
Bird.momentum = 20
Perte de vitesse = -0.5
```

5.7 Les formes et leur sprite

```
Bird.java
Pipe.java
PipeCouple.java
```

Pour afficher une image (un sprite) on crée une ImageView, dans laquelle on met une image.

```
new ImageView(new Image("Sprites/cloudbg1.png"));
```

Pour faire bouger le sprite avec la forme à qui il appartient, on fait en sorte qu'à chaque fois que ladite forme bouge, le centre du sprite suit le centre de la forme.

Ici, lorsqu'un couple de tuyau bouge :

```
public void move(int speed) {
     this.pipe1.moveLeft(speed);
     this.pipe1.refreshPipeSprite();
     this.pipe2.moveLeft(speed);
     this.pipe2.refreshPipeSprite();
}
```

5.8 La detection des touches

```
Main.java
```

La détection des touches se fait de manière très simple :

```
scene.setOnKeyPressed(event -> {
    KeyCode keyCode = event.getCode();
    if (keyCode.equals(KeyCode.SPACE)) {
        //Faire quelque chose si SPACE est appuyé
    }
});
```

Une simple fonction qui écoute les touches du clavier et qui permet de détecter lorsqu'une touche est appuyée.

La fonction doit être placée dans le `start`, pour pouvoir accéder à la `scene`

Si l'on souhaite détecter lorsque la touche est relâchée, on remplace `SetOnKeyPressed` par `SetOnKeyRelease`.

Pour changer de touche, on modifie le `KeyCode`.

Pour n'effectuer qu'une seul fois une action, j'ai souvent mis en place des booléen qui dès que l'action est faite une fois, ne la laisse plus se reproduire tant que la touche n'a pas été relâchée.

6. Sources

- Gluon : https://gluonhq.com/products/javafx/
- Classe Rectangle : https://docs.oracle.com/javase/8/javafx/api/javafx/scene/shape/Rectangle.html
- Détection des touches : https://stackoverflow.com/questions/37472273/detect-single-key-press-in-javafx
- Animation timer : https://stackoverflow.com/questions/50337303/how-do-i-change-the-speed-of-an-animationtimer-in-javafx
- Font javafx : https://www.tutorialspoint.com/javafx/javafx_text.htm#:~:text=You%20can%20change%20the%20font,scene.
- Rotation d'un imageView : https://stackoverflow.com/questions/34166627/javafx-rotate-imageview
- Inspiration pour sauts fluides : https://stackoverflow.com/questions/41314228/javafx-creating-smooth-animation-without-creating-a-new-thread
- Le jeu en ligne : http://flappybird.io/
- Précédents exercices réalisés en module