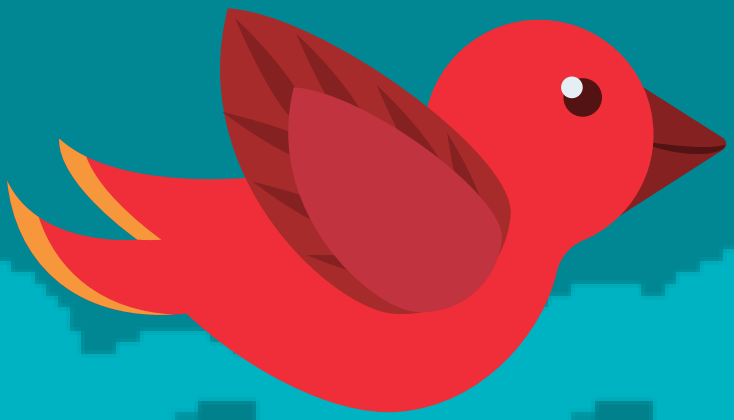
Projet PCII
Flappy Bird

Rapport de projet
2021-2022



Merand Yoann - Miage

Sommaire

I. Introduction

Dans le cadre du module PCII, nous devons créer une application inspirée du jeu Flappy Bird qui respecte les principes de celui-ci en version quelque peu simplifié.

Le principe du jeu ici est le suivant. Nous avons un ovale affiché à l'écran avec une ligne brisée.
Le but de l'utilisateur est d'empêcher les extrémités de l'ovale de toucher la courbe ou la partie est terminée.
Il faut savoir que l'ovale se déplace vers le haut lorsque nous avons une interaction avec celui, autrement, il se déplace vers le bas.

Nous devons donc incorporer une interface graphique à notre projet. Voici ci-dessus, un exemple d'interface du jeu.

II. Analyse Globale

Pour le développement de cette application, il existe cinq fonctionnalités principales. Nous devions implémenter l'interface graphique avec l'ovale et la ligne brisée. Ensuite, nous devions faire un défilement automatique de la ligne brisée, la réaction de l'ovale face au clic ou à l'interaction clavier de l'utilisateur. La détection de collisions, ainsi que l'ajout d'un oiseau animé en arrière-plan.

Pour ce faire, nous avions plusieurs séances afin d'avancer au mieux dans le projet.
Pour la première séance, nous avions les fonctionnalités les plus simples à effectuer.

- L'implémentation de l'interface graphique de l'ovale
- Déplacement de l'ovale face aux interactions de l'utilisateur

Pour la seconde séance, nous avons commencé les Threads pour rendre notre projet plus dynamique.

- -Implémentation de la ligne brisée
- Mouvement de la ligne brisée

Enfin, lors de la troisième séance, il a fallu rendre le jeu plus "attrayant" de ce fait nous avons ajouté des fonctionnalités pour l'utilisateur lorsqu'il joue.

- Ajout d'un background
- Ajout d'un oiseau qui vole (en fond)
- Ajout de différentes bandes sons
- Ainsi qu'un score qui rend le jeu plus ludique

III. Plan de Développement

Diagramme De Gantt

IV. Conception Générale

Nous avons décidé d'utiliser le modèle MVC pour le développement de notre interface graphique.

Pourquoi le Modèle MVC ?

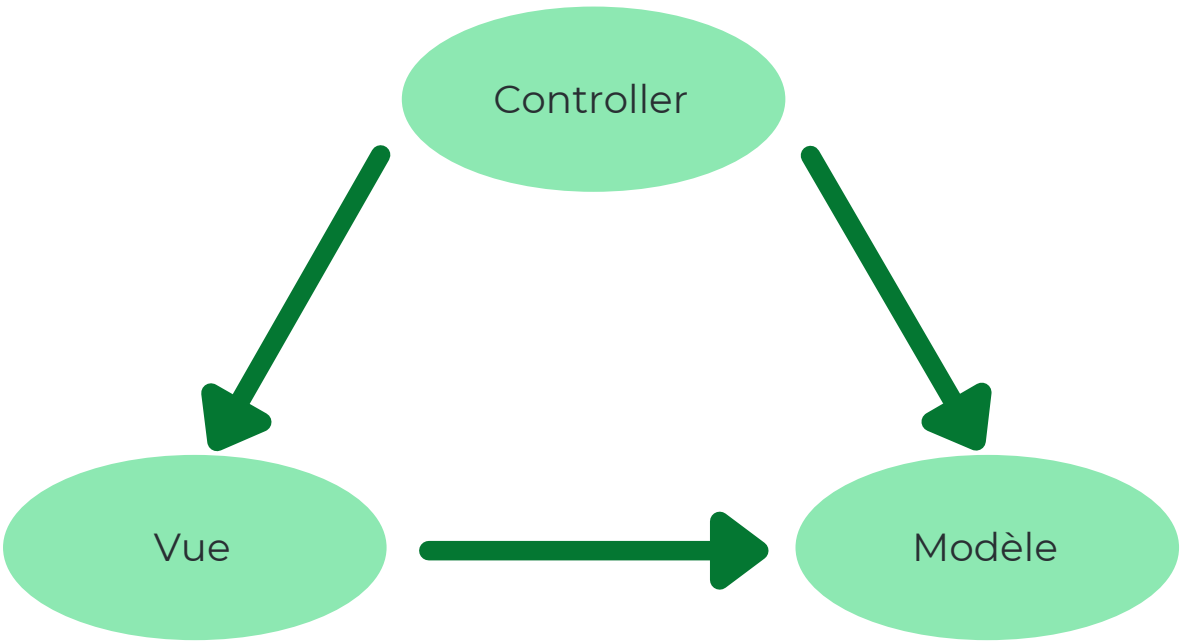
L'avantage du Modèle MVC, est la structure du code. Chaque classe a un rôle distinct. De plus, cela permet une plus grande souplesse et de simplicité dans le développement de l'application. De plus, ayant une interface graphique dans le projet, le modèle MVC semble plutot évident puisque il permet d'avoir un classe Vue

Le modèle MVC (Modèle, Vue, Controller) peut se définir de la manière suivante :

Modèle définit l'ensemble des données qui caractérisent l'état de votre interface. La modification de ces données correspond à un changement de l'affichage dans l'interface graphique.

Vue définit l'affichage : comment l'état du modèle est rendu visible à l'utilisateur.

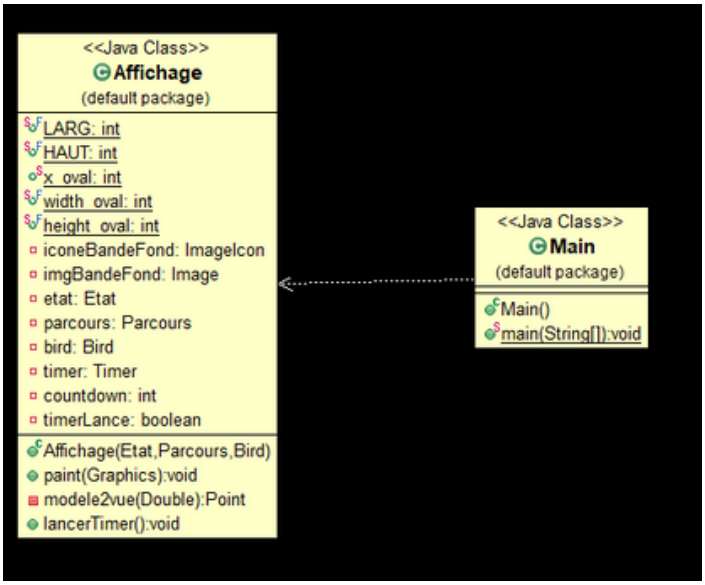
Contrôleur définit la manière dont l'état du modèle change : c'est ce qui fait le lien entre votre interface graphique et le reste du monde. Il effectue les changements dans le modèle et informe la vue d'un changement. Lorsque l'interface est interactive, c'est aussi lui qui gère les événements.

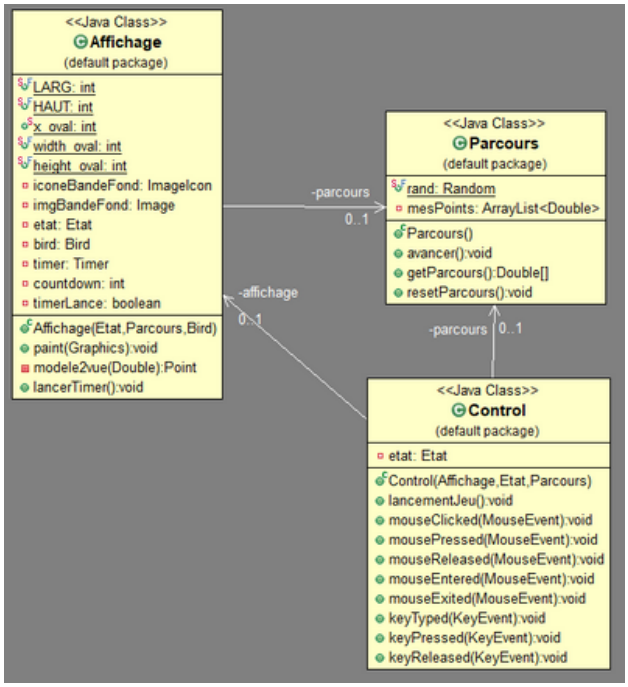


V. Conception Détaillée

Voyons à présent la conception détaillée de l'application. Pour ce projet, j'utilise l'API Java Swing.
Commençons par la première fonctionnalité : l'interface graphique avec l'ovale et la ligne brisée.
Comme on peut le voir ici, la classe affichage affiche permet à elle seule d'afficher tout ce qui apparaît dans l'interface graphique.

Pour l'ovale, la classe affichage défini sa longueur (height_oval) ainsi que sa largeur (witdh_oval).
De plus la fonction "paint" permet de dessiner tout cela, et la classe main permet de lancer le programme.

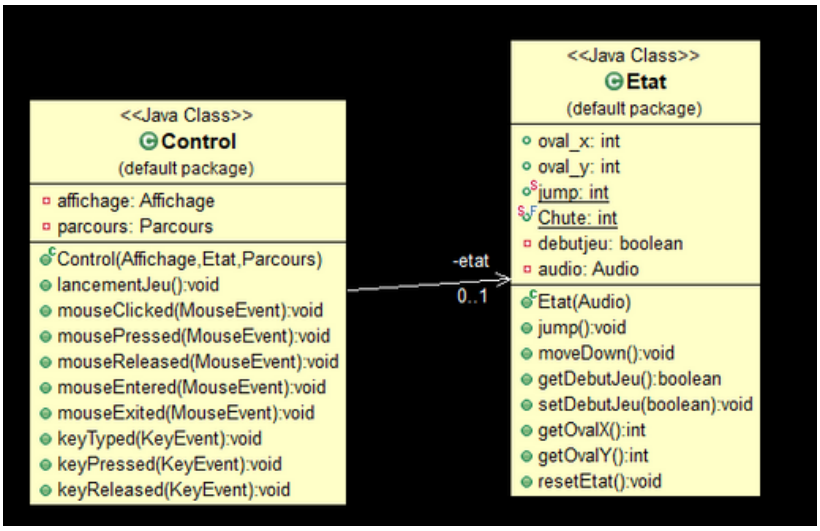


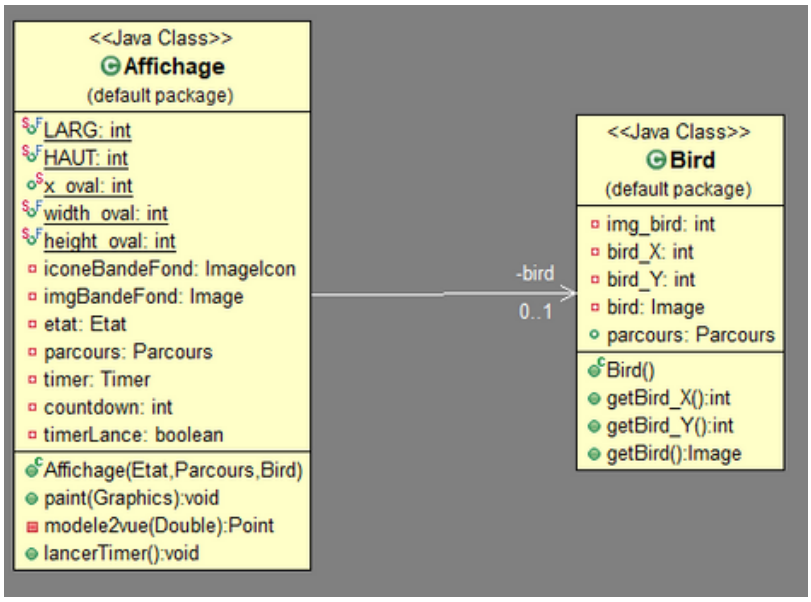


Pour cette deuxième fonctionnalité, il s'agit du défilement automatique de la ligne brisée.
Ici, la fonction avancer dans la classe parcours permet de générer de nouveaux points dans une array list.
La fonction Avancer est appelé dans le Thread de la classe Control, ce qui permet à la courbe de se déplacer à l'infini puisqu'elle génère de nouveaux points.
Enfin la méthode Modele2vue convertie l'abscisse et l'ordonnée de la ligne brisée pour qu'elle soit adaptée à la fenêtre du jeu

V. Conception Détaillée

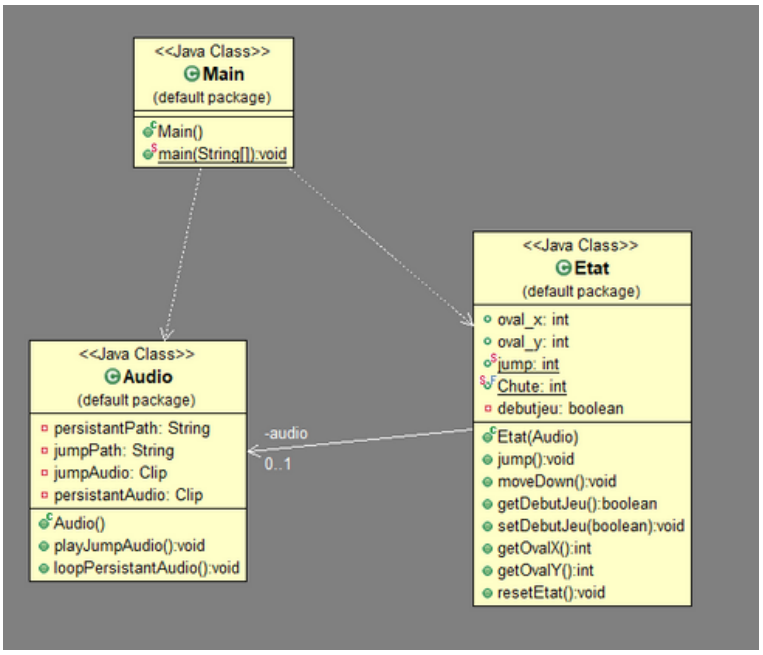
Pour cette troisième fonctionnalité, il s'agit du déplacement de l'ovale dans l'interface graphique.
Ici, la classe Etat, correspond à l'état de l'ovale. La fonction mooveDown dans celle-ci permet de faire chuter l'ovale. Et la méthode jump permet de faire sauter l'ovale à chaque interaction du clavier ou de la souris.
Et pour ces interactions, nous passons dans la classe Controlleur ou nous avons ici KeyPressed ainsi que MouseClicked





Pour cette cinquième fonctionnalité, j'ai implémenté un oiseau en élément de décors.
Cet oiseau est au départ un gif découpé en plusieurs images et converti en "png".
La classe Bird permet de récupérer la position en X et en Y pour les récupérer grâce à des getteur dans la fonction Paint de la classe affichage.

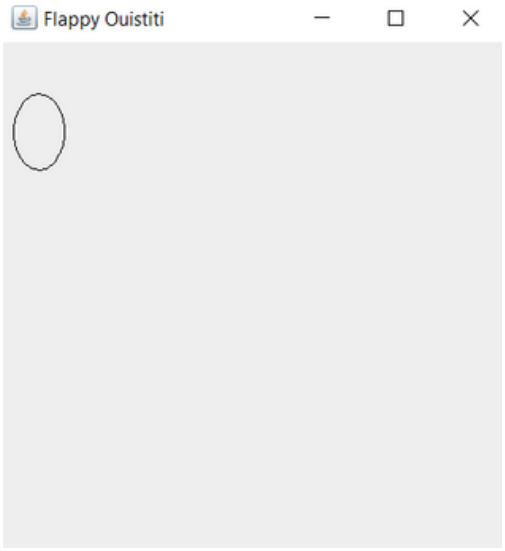
V. Conception Détaillée



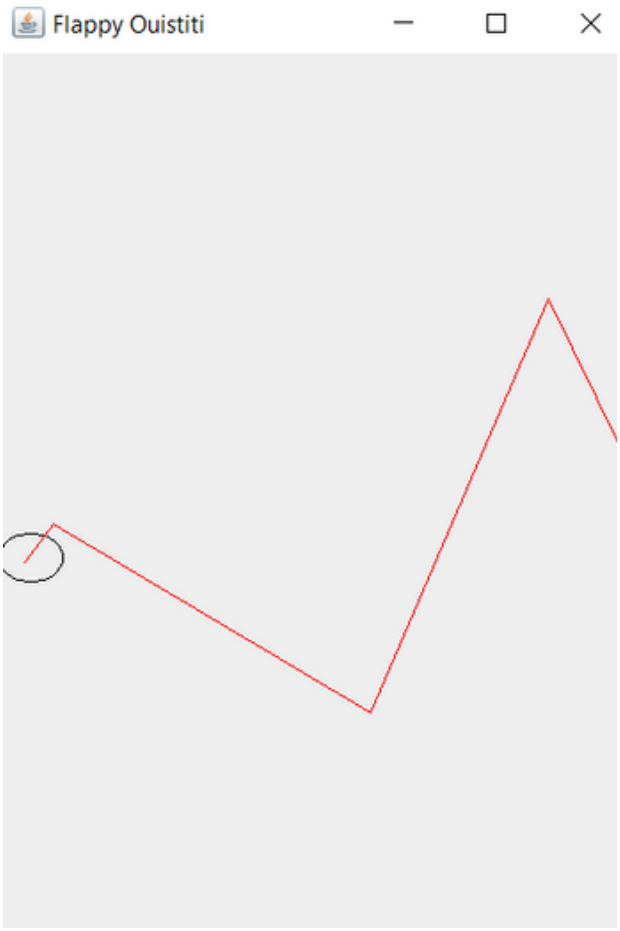
En autre fonctionnalité, j'ai décidé d'implémenter une bande son au projet. Les deux bandes sons que j'ai ajoutés sont une musique de fond et un effet sonore à chaque jump() de l'ovale.

VI. Résultat

Rendu Séance 1 :



Rendu Séance 2 :



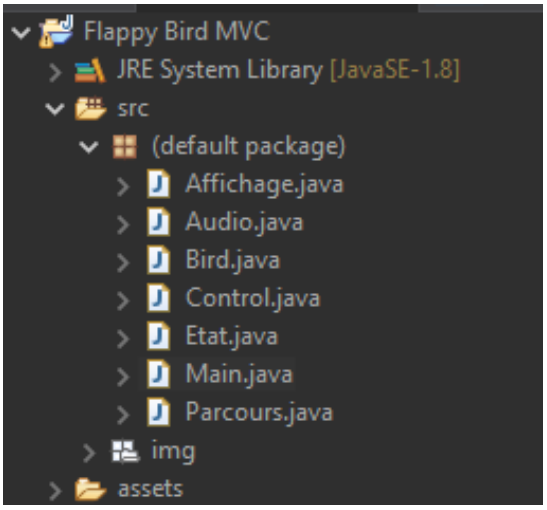
Rendu Séance 3 :

VII. Documentation utilisateur

Si vous êtes un utilisateur du projet, voici comment procéder pour exécuter le projet.

Il vous faut tout d'abord avoir un IDE avec Java ou avoir simplement Java d'installer sur son ordinateur.

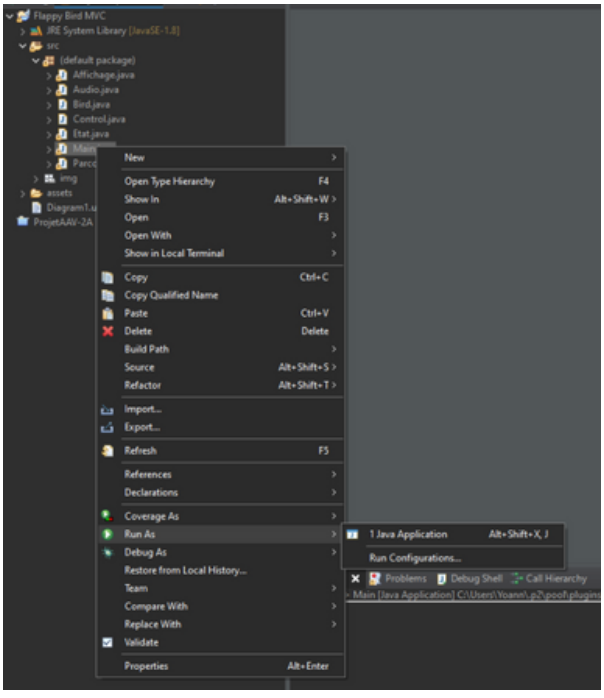
Pour exécuter le code, vous avez deux possibilités.
La première, la plus compliquée est de :
- Lancer votre IDE,
- Ouvrir le projet "Flappy Bird"
- Sélectionner la classe Main
- Puis cliquer sur "1 Java Application
- Vous n'avez plus qu'a jouer !



La seconde méthode, beaucoup plus simple

- Sélectionner Flappy Bird.jar
- Double-cliquer et le jeu apparaît !

Le seul inconvénient à cette deuxième méthode est que la bande audio de l'application ne démarre pas.



VIII. Documentation Développeur

En tant que développeur, si vous souhaitez modifier le projet, voici les choses qui pourront vous aider.

Dans ce projet, j'utilise le modèle MVC de ce fait, la classe vue regroupe tout l'affichage de l'interface graphique et plus en particulier la fonction "paint". Qui permet de dessiner tout ce qui apparaît sur l'interface Graphique.

Les classes les plus importantes pour la conception du jeu sont parcours, Etat et Controller. La méthode main est dans la classe "Main".

Pour modifier le code, les principales constantes à modifier sont la taille de la fenêtre dans la classe Affichage ainsi que la taille de l'ovale qui sont dans la même classe.

Je n'ai cependant pas pu implémenter la collision dans le jeu et ceci est un point très important dans le projet. De plus, j'aimerais ajouter une autre bande audio à chaque fois que l'utilisateur perd la partie.

Enfin, un autre axe d'amélioration possible, serait de créer différents niveaux de difficultés avec une ligne brisée plus ou moins ardu et la vitesse de la chute qui augmente drastiquement.

IX. Conclusion et perspectives

Pour conclure ce rapport, l'application fonctionne bien dans l'ensemble et respecte la plupart des points demandés dans le sujet.
Cependant, il n'a pas pu être totalement finalisé.

Pour revenir à ce qui a été compliqué pour ma part, c'est avant tout la création de la ligne brisée car ayant très peu d'expérience en java Swing, il m'a fallu regarder la java doc en détails pour comprendre bien le fonctionnement de certains imports comme "import java.awt.geom.Point2D.Double".

La principale difficulté dans ce projet, vient de la collision, car il faut savoir lorsque l'ordonnée rentre en contact avec la ligne brisée.
De plus ayant un problème pour la collision, le fait de faire un score en devient impossible.
De ce fait, je suis parti sur un système différent en partant sur une idée de chronomètre.
J'ai donc fait un compte à rebours de 60 secondes qui réinitialise la partie quand le chronomètre est fini.

Il reste cependant beaucoup d'axes d'amélioration possible pour le projet. La première serait de rendre plus fluide l'affichage et également d'avoir une ligne brisée plus arrondi.
J'ai également pu imaginer devoir déplacer plusieurs ovales simultanément ainsi que de créer différents niveaux de difficultés.

Dans l'ensemble de ce projet, la découverte de java swing m'a paru très intéressante ainsi que d'avoir un affichage graphique.