

|  |  |
| --- | --- |
|  |  |

**Département Informatique**

**Licence 3e année**

**Conception et développement d’un jeu**

ROQUINARC’H Hoël

Année universitaire 2020-2021

Table des matières

[Introduction 3](#_Toc93216745)

[Analyse 3](#_Toc93216746)

[Plan de développement 4](#_Toc93216747)

[Conception générale 4](#_Toc93216748)

[Conception détaillée 5](#_Toc93216749)

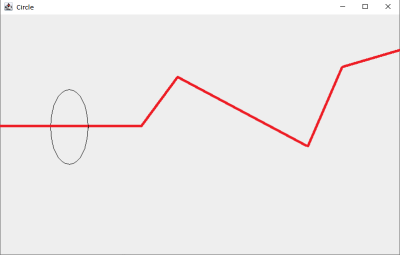
[Documentation utilisateur 5](#_Toc93216750)

[Documentation développeur 5](#_Toc93216751)

[Conclusion et perspectives 5](#_Toc93216752)

# Introduction

Le but de ce projet est la réalisation d’un mini-jeu. Inspiré par le jeu populaire Flappy-bird, l’interface de notre jeu sera constituée d’un ovale qui descendra continuellement le long d’une ligne horizontale avec des reliefs. A chaque clic, l’ovale remontera légèrement et le joueur perd lorsque ce dernier touche la ligne. L’interface de notre jeu ressemblera à quelque chose comme cela :



# Analyse

Le jeu peut être divisé en 3 fonctionnalités :

* L’interface graphique comprenant l’ovale mais générant et affichant aussi la ligne brisée avec des reliefs aléatoires.
* Le défilement automatique de la ligne.
* L’interaction du « saut » de l’ovale a chaque clic de l’utilisateur.

Nous nous concentrerons tout d’abord à ces sous-fonctionnalités :

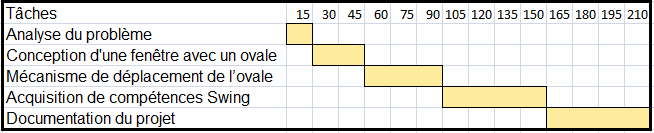
* La création de la fenêtre avec l’ovale.
* Le déplacement de l’ovale lors des clics du joueur.

Nous nous concentrerons sur ces sous-fonctionnalités car elles sont importantes, elles sont faciles à réaliser et elles forment une bonne base pur notre projet.

# Plan de développement

Liste des tâches :

* Analyse du problème (15 mn)
* Conception, développement et test d’une fenêtre avec un ovale (30 mn)
* Conception, développement et test du mécanisme de déplacement de l’ovale (45 mn)
* Acquisition de compétences en Swing (60 mn)
* Documentation du projet (60 mn)



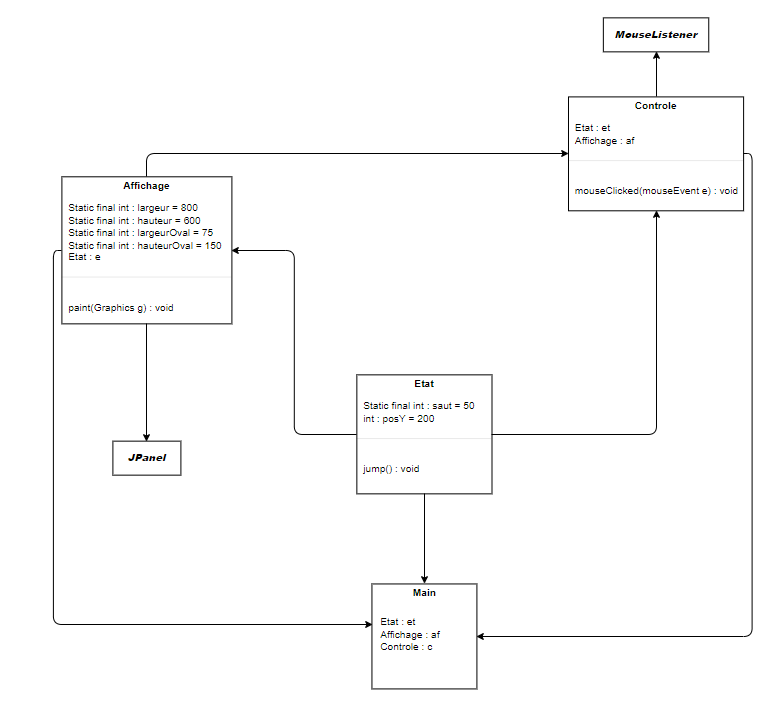
# ****Conception générale****

Nous avons décidé d’utiliser le modèle MVC :

* Dans la partie modèle, on inclura le code qui enregistre la position de l’ovale et effectue le « saut ».
* Dans la partie vue, on mettra le code qui dessine et met à jour l’affichage du jeu.
* Enfin dans la partie contrôleur, on ajoutera la partie de code permettant la détection du clic de l’utilisateur.

# Conception détaillée

Pour la fenêtre avec un ovale, nous utilisons l’API Swing et la classe JPanel. Nous définissons les dimensions de l’ovale et de la fenêtre dans des constantes. Pour le déplacement de l’ovale, nous utilisons la programmation événementielle avec la classe mouseListener et la hauteur est définie dans une constante.



Puis, nous avons créé 2 nouvelles classes afin de créer une ligne brisée.

La classe parcours permet de crée et afficher la ligne brisée grace a une liste de coordonnées. Elle crée au fur et a mesure que l’ovale avance afin de ne pas surcharger le tableau de valeur avec des valeurs inutiles.

La classe avancer permet de décaler la ligne brisée pour que cette dernière paraisse en mouvement.

Nous avons ensuite séparé le fonctionnement de l’application en trois threads différents :

* Le premier gère la chute de l’ovale
* Le deuxième gère les saut lorsque l’utilisateur clique
* Enfin, le dernier, s’occupe de faire « avancer » l’ovale tout en générant la ligne brisée.

Documentation utilisateur

Pré requis : Java avec un IDE (ou Java tout seul si vous avez fait un export en .jar exécutable)

Mode d’emploi (cas IDE) : Importez le projet dans votre IDE, sélectionnez la classe Main à la racine du projet puis « Run as Java Application ». Cliquez sur la fenêtre pour faire monter l’ovale.

Mode d’emploi (cas .jar executable) : double-cliquez sur l’icône du fichier .jar. Cliquez sur la fenêtre pour faire monter l’ovale.

# ****Documentation développeur****

Les prochaines fonctions que nous devrons réaliser seront :

* La création de la ligne brisée.
* Le défilement de cette dernière pour donner la sensation que l’ovale s’y déplace.

# ****Conclusion et perspectives****

Pour conclure, nous avons terminé les fonctions dites simples du projet énoncées plus tôt.

Le principal problème rencontré est l’implémentation de la collision du cercle. Avec un peu plus de temps cette dernière serait réalisable. la séparation du code déjà existant entre les différentes parties du modèle MVC aussi a posé problème.

Finalement, il ne nous restera que la fonction de collision à faire pour conclure le projet.