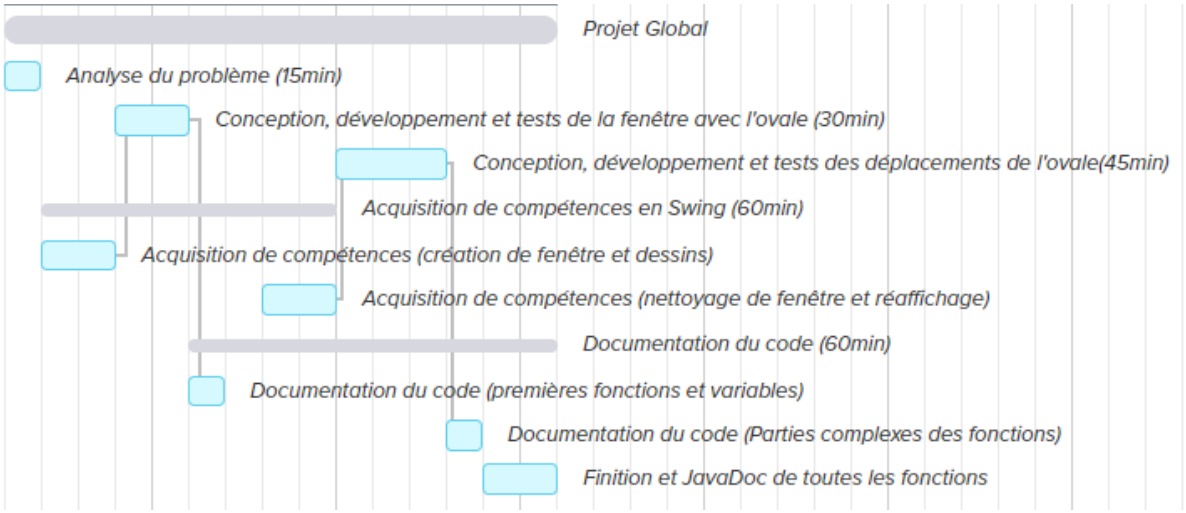
Rapport de projet n°1

Nathan Monsoro

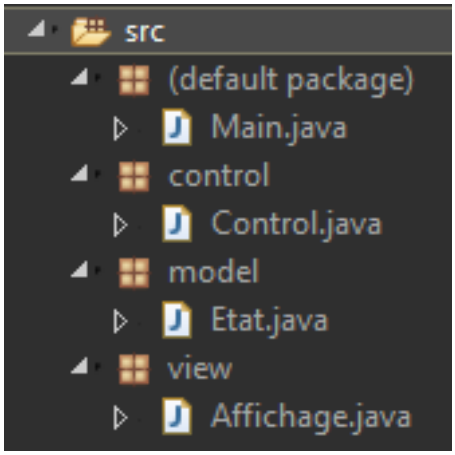
L'objectif technique de ce projet est de nous familiariser avec la structure MVC. Nous avons donc pour but de créer un mini jeu semblable au bien connu Flappy Bird. Le principe de notre projet sera le suivant : un ovale qui monte et descend, devra rester sur une ligne brisée qui défile. Pour faire monter l'ovale, l'utilisateur n'aura qu'a cliquer avec la souris dans la fenêtre du jeu. L'ovale redescendra tout seul au cours du temps. Pour réaliser cela, on commencera par se familiariser avec les classes JFrame et JPanel d'abord créer une fenêtre vide, puis avec un ovale statique. On devra ensuite utiliser la classe MouseListener pour que notre programme réagisse aux clics souris. Grâce à cela, nous pourront faire monter notre ovale avec la souris.

Nous pouvons identifier trois groupes principaux de fonctionnalités dans ce projet de mini-jeu : d'abord l'interface graphique qui affichera l'ovale ainsi que la ligne brisée que ce dernier devra essayer de suivre dans la fenêtre, puis le défilement automatique de la ligne brisée à l'aide d'une horloge, et enfin les déplacements de l'ovale. Les déplacements de l'ovale sont soit des réactions aux clics souris de l'utilisateurs pour le faire monter, soit une descente automatique lorsqu'on ne le fait plus monter pendant un temps. Lors de cette séance, on ne s'est occupé que des sous fonctionnalités qui suivent. Dans un premier temps la création de la fenêtre avec le dessin de l'ovale, puis le déplacement de l'ovale vers le haut lorsque l'utilisateur fait un clic à la souris dans la fenêtre du jeu. Ces deux fonctionnalités sont sûrement les plus simples du mini-jeu, ce qui en font de bonnes premières implémentations pour découvrir le modèle MVC. Évidemment, l'étape de création de la fenêtre bien qui pas spécialement compliquée est capitale dans notre projet, puisque sans cela, aucun espace pour afficher notre jeu, ni pour interagir avec l'utilisateur. Il nous faut ensuite dans cette fenêtre dessiner l'ovale. Cette étape est aussi très simple, mais importante aussi puisque l'ovale est l'élément que le joueur va le plus regarder, puisque c'est sa position par rapport à la ligne brisée qui va définir s'il est en train de gagner ou en mauvaise posture. Cet ovale représente le joueur. La fonctionnalité qui fait « sauter » l'ovale est aussi très importante, puisque faire monter l'ovale est la seule interaction que l'utilisateur peut avoir avec le jeu. C'est donc par la seule commande du jeu, le clic souris, que l'utilisateur devra se déplacer de son mieux pour gagner.

Mon temps de travail s'est à peu près réparti comme suit : dans un premier temps j'ai pris environ 15 minutes à analyser le problème, les compétences que j'avais à acquérir et à réutiliser et se que donnerait le projet final. J'ai ensuite réfléchi à la conception et au développement de la première fonctionnalité que nous avons implanté, qui est la fenêtre qui affiche l'ovale. J'ai aussi fait plusieurs tests sur cette fenêtre avec différentes dimensions et couleurs pour l'ovale. Cette étape m'a pris environ 30 minutes. Après cela, j'ai passé environ 45 minutes à la conception, au développement et aux tests du mécanisme de déplacement de l'ovale. Cette fonctionnalité est capitale, et le sera encore plus tard dans le projet, je me suis donc appliqué à la faire le mieux possible. Tout au long du projet, il m'a fallu apprendre à utiliser la bibliothèque Swing, afin d'utiliser les bonnes fonctions aux bons moments. Cela m'a permis aussi de réinvestir quelques vagues connaissances que j'en avais qui dataient des années passées. Mis bout à bout, tout cet apprentissage réinvestissement de connaissance m'a pris environ 1 heure. Finalement, j'ai mis environ autant de temps à documenter mon projet. La documentation fu très longue a cause des nombreuses nouvelles notions et méthodes que j'ai utilisées dans le code, d'où le fait que j'ai mis 60 minutes environ à la faire.



Ci-dessus le diagramme de Gantt qui représente la répartition de mon temps sur les 3h30 de la séance

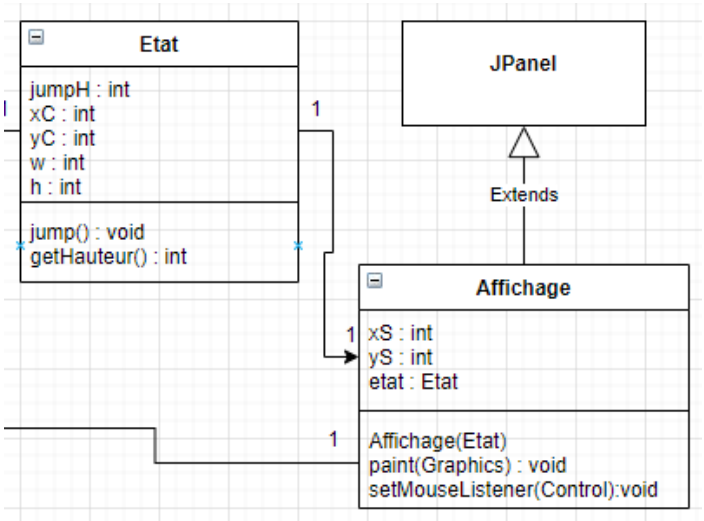


Lors de ce projet, nous avons dû nous familiariser avec une architecture de code toute nouvelle pour nous, le modèle MVC. (Vous trouverez ci-contre l'organisation en package des différentes classes du projet.) Nous avons donc organisé notre code dans 3 packages différents comme ci-contre, et laissé la classe Main dans le package de base. La classe Etat représente l'état du jeu à afficher, elle contrôle la position de l'ovale. La classe Control gère les entrées de l'utilisateur, elle appelle la fonction de saut de la classe état quand l'utilisateur clique avec la souris, et appelle la fonction qui met à jour l'affichage dans la fenêtre. Enfin la classe Affichage s'occupe d'afficher l'état du jeu. Elle crée la fenêtre et dessine l'ovale.

Nous allons dans cette partie parler de façon plus détaillée de l'implémentation des fonctionnalités.

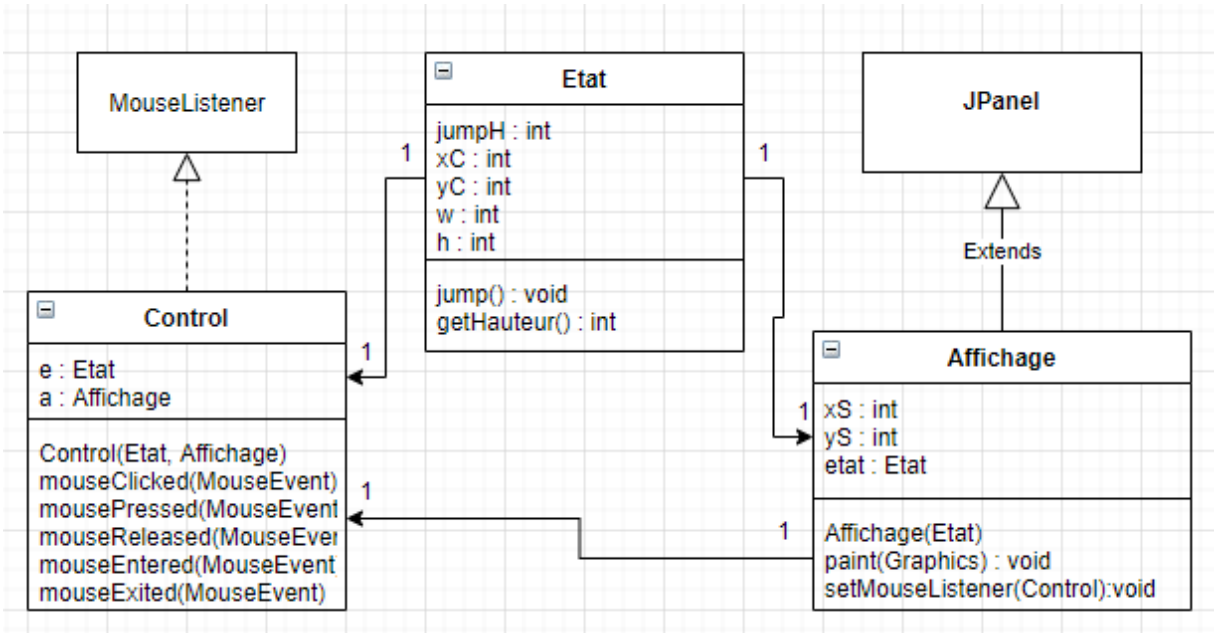
D'abord l'affichage de la fenêtre et de l'ovale. Pour cette fonctionnalité on utilise l'API Swing. On a aussi besoin de la classe Affichage qui hérite de JPanel, et de la classe Etat. Les dimensions de la fenêtre que l'on veut créer sont stockées dans les constantes xS et yS de la classe Affichage. Les dimensions de l'ovale sont, elles, dans les constantes xC, yC, w et h de la classe Etat. Pour afficher la fenêtre on crée directement dans la classe Main une JFrame auquel on ajoute l'affichage. Grâce à l'attribut etat de la classe, on a relié l'état du jeu à notre affichage.

Ci-dessous le diagramme des classes Etat et Affichage.

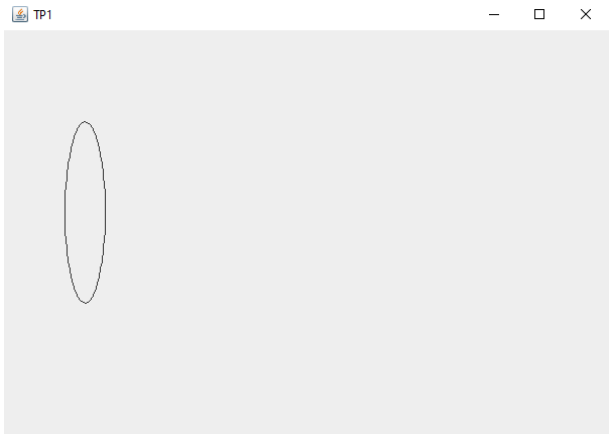


Nous avons aussi implémenté le déplacement de l'ovale sur l'axe horizontale dans la fenêtre. La hauteur d'un saut est définie dans la classe Etat par la constante jumpH. La fonction jump() qui fait monter notre ovale est appelé dans la classe Control qui implémente la classe MouseListener, lorsque l'utilisateur fait un clic de souris dans la fenêtre. Après cela, le contrôleur demande à la classe Affichage de mettre à jour la fenêtre avec la fonction repaint(). Dans la fonction jump(), on vérifie si l'ovale ne sortirait pas du cadre après le saut, si c'est le cas, on le colle simplement à la bordure supérieure de la fenêtre, sinon on le fait monter de la valeur de jumpH.

Ci-dessous le diagramme des 3 classes de notre projet.



Voici ci-dessous ce que donne notre projet dans l'état actuel. On voit bien l'ovale dans la fenêtre. Il n'en sort pas et monte quand on clique à la souris.



¨Pour exécuter le code, il vous faudra un IDE avec Java pour l'exécuter à partir du code, ou bien Java pour l'exécuter à partir d'un fichier .jar.

A partir d'un IDE, importez simplement le projet, et à partir de la classe Main faites « Run as Java Application ». La fenêtre avec l'ovale s'ouvrira alors, vous pourrez faire monter ce dernier en cliquant dans la fenêtre.

A partir du fichier .jar, double cliquez dessus pour l'exécuter et ouvrir la fenêtre. Vous pourrez faire monter l'ovale de la même façon en cliquant dans la fenêtre.

Pour continuer ce projet, on ajoutera la ligne brisée et son défilement automatique qui donnera l'impression que l'ovale progresse vers la droite.

A l'heure actuelle, nous avons codé la fenêtre et l'ovale qui s'y déplacera. Nous avons pu nous familiariser avec l'organisation du format MVC et de la bibliothèque Swing. Il nous reste à implémenter la ligne brisée que l'ovale devra suivre, et son défilement pour que l'utilisateur ai l'impression que l'ovale avance.