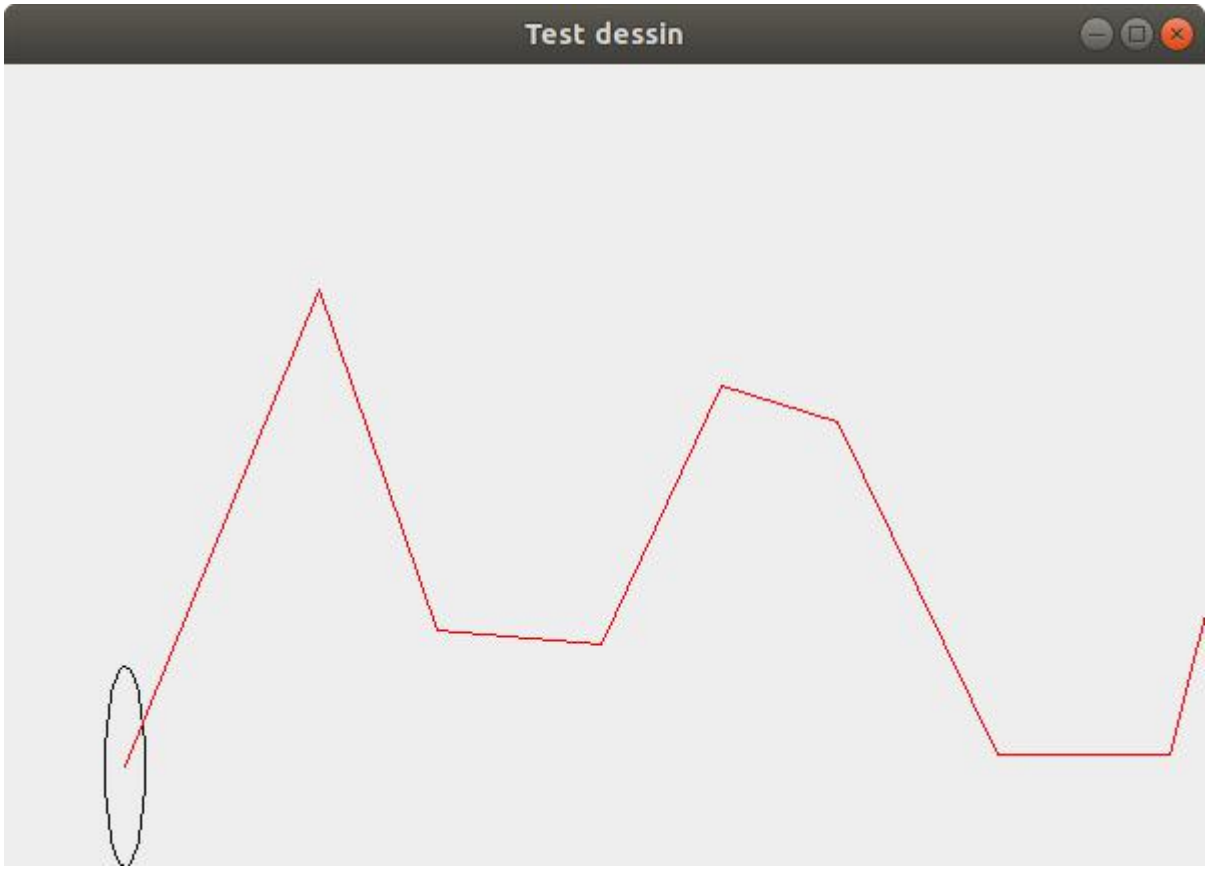
Rapport du développement du tutoriel

HUANG Shiqing

Introduction

Nous voulons réaliser un mini-jeu inspiré de flappy bird dans lequel un ovale se déplace le long d'une ligne brisée. Le but du jeu est d'éviter que l'ovale sorte de la ligne. Pour cela, le joueur peut cliquer sur l'écran pour faire monter l'ovale, qui redescend ensuite tout seul. Voici à quoi pourrait ressembler l'interface graphique de notre jeu:



Analyse globale

Il y a trois fonctionnalités principales:

1. l'interface graphique avec l'ovale et la ligne brisée
2. le défilement automatique de la ligne brisée
3. la réaction de l'ovale aux clics de l'utilisateur

Dans la première partie du projet (séance 1), nous nous intéressons uniquement à un sous-ensemble de fonctionnalités:

- création d'une fenêtre dans laquelle est dessiné l'ovale ;
- déplacement de l'ovale vers le haut lorsqu'on clique dans l'interface.

Ces deux sous-fonctionnalités sont prioritaires et simples à réaliser.

Dans la deuxième partie du projet (séance 2), nous nous intéressons au sous-ensemble de fonctionnalités:

- l'ovale perde de l'altitude lorsque l'utilisateur ne clique pas sur l'écran;
- Dessiner la ligne brisée dans la fenêtre
- défilement automatique de la ligne brisée

Dans la troisième partie du projet (séance 3) pour avoir un système fonctionnel est de détecter les collisions.

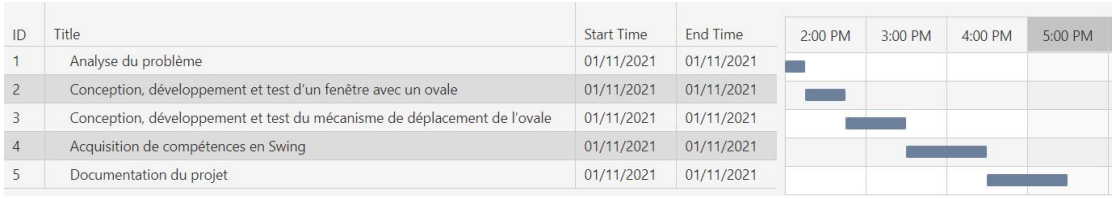
Il y a deux améliorations supplémentaires:

- Amélioration des dessins
- Ajout d'éléments de décors

Plan de développement

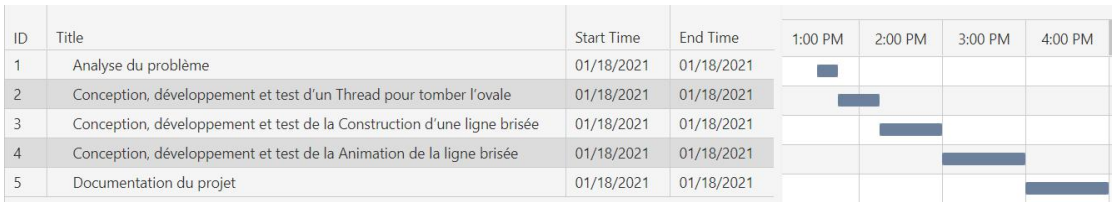
la deuxième partie du projet (séance 1)
Liste des tâches :

Analyse du problème (15 mn)
Conception, développement et test d'un 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)



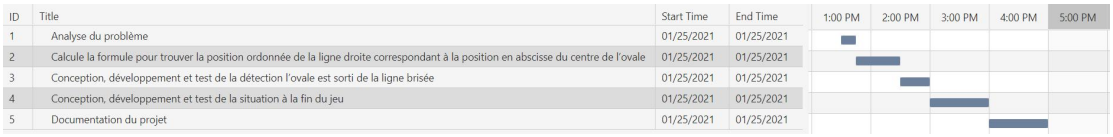
la deuxième partie du projet (séance 2)
Liste des tâches :

Analyse du problème (15 mn)
Conception, développement et test d'un Thread pour tomber l'ovale (30 mn)
Conception, développement et test de la Construction d'une ligne brisée (45 mn)
Conception, développement et test de la Animation de la ligne brisée(60 mn)
Documentation du projet (60 mn)



la troisième partie du projet (séance 3)
Liste des tâches :

Analyse du problème (15 mn)
Calcule la formule pour trouver la position ordonnée de la ligne droite correspondant à la position en abscisse du centre de l'ovale (45 mn)
Conception, développement et test de la détection l'ovale est sorti de la ligne brisée (30 mn)
Conception, développement et test de la situation à la fin du jeu(60 mn)
Documentation du projet (60 mn)



ID	Title	Start Time	End Time
1	Analyse du problème	01/25/2021	01/25/2021
2	Calcule la formule pour trouver la position ordonnée de la ligne droite correspondant à la position en abscisse du centre de l'ovale	01/25/2021	01/25/2021
3	Conception, développement et test de la détection l'ovale est sorti de la ligne brisée	01/25/2021	01/25/2021
4	Conception, développement et test de la situation à la fin du jeu	01/25/2021	01/25/2021
5	Documentation du projet	01/25/2021	01/25/2021

Conception générale

L'interface graphique s'est construite autour du modèle MVC.

La classe Affichage s'occupe de dessiner l'interface.

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

La classe controller effectue les changements dans l'état et informe la vue d'un changement. Lorsqu' on clique sur l'écran, c'est lui qui gère les événements.

La classe Voler est un Thread qui appelle une fonction de l'état et change l'affichage dans l' interface graphique, il est exécuté de manière entrelacée par le processeur.

La classe Parcours définit l'ensemble des données(les points) de la ligne brisée.

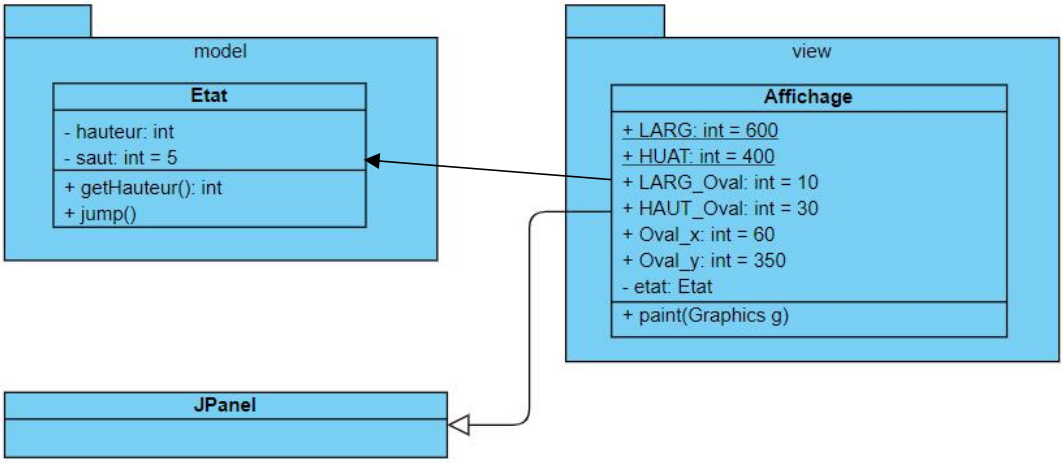
La classe Avancer est un Thread qui appelle une fonction deu parcours et change l'affichage dans l'interface graphique, il est exécuté de manière entrelacée par le processeur.

Conception détaillée

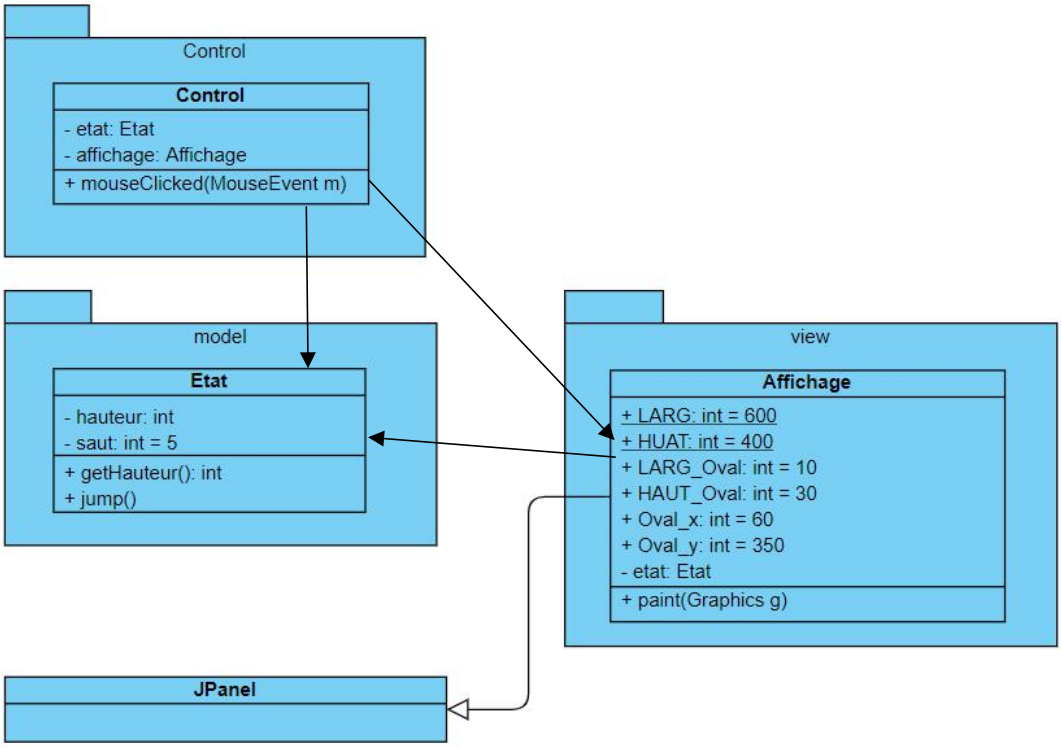
1.1 Pour la fenêtre avec un ovale, nous utilisons l'API Swing et la classe JPanel. Nous définissons les dimensions de l'oval et de la fenêtre dans des constantes

```
/** Constantes */
/* Les dimensions du JPanel (largeur et hauteur) */
public static final int LARG = 600;
public static final int HAUT = 400;
```

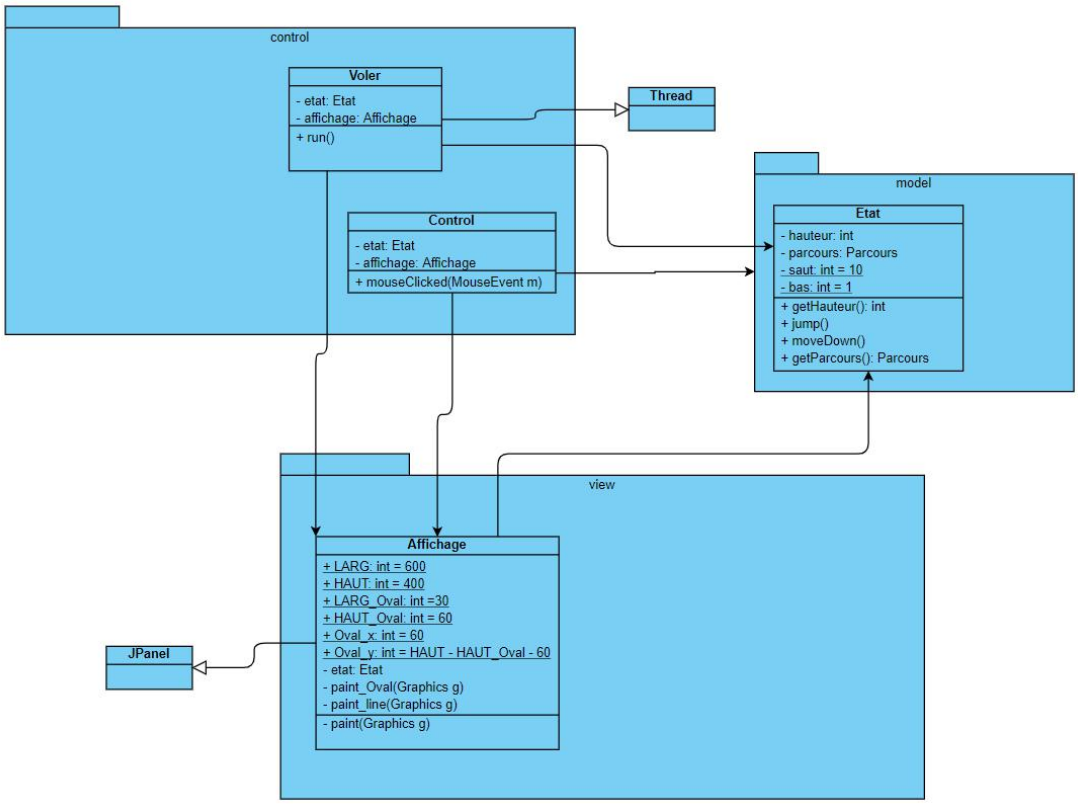
```
/** Les dimensions initiales du ovale */
/* largeur */
public static final int LARG_Oval = 10;
/* hauteur */
public static final int HAUT_Oval = 30;
/* centre */
public static final int Oval_x = 60;
public static final int Oval  y = 350;
```



1.2 Pour le déplacement de l'ovale, nous utilisons la programmation événementielle avec la classe MouseListener et la hauteur est définie dans une constante.



2.1 Pour l'ovale perde de l'altitude lorsque l'utilisateur ne clique pas sur l'écran, on ajoute une méthode moveDown qui permet de modifier la valeur de la hauteur de quelques pixels vers le bas (définie dans une constante). Ensuites, on créer une classe Voler qui hérite de Thread, La méthode run de ce thread utilisera une boucle infinie pour appelle moveDown.

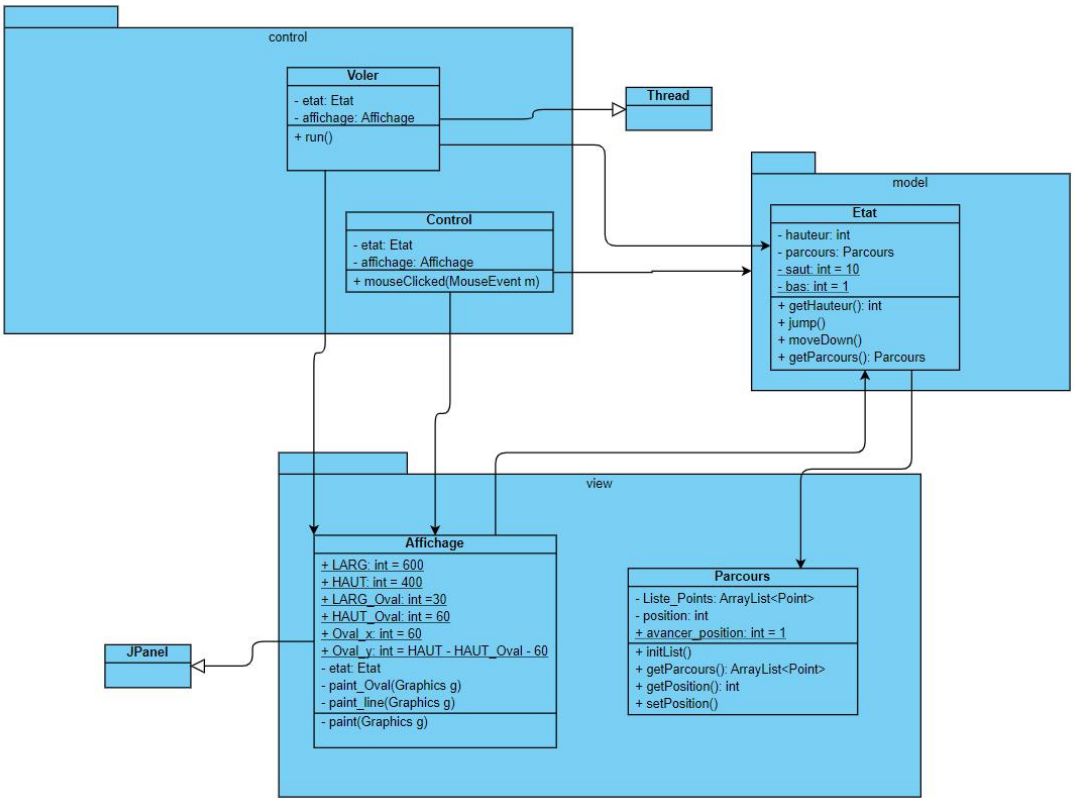


2.2 Pour Dessiner la ligne brisée dans la fenêtre, on construise une classe Parcours dont l'attribut principal est un objet ArrayList<Point> (une liste de points) et utilise drawLine dans la classe Affichage pour afficher la ligne brisée.

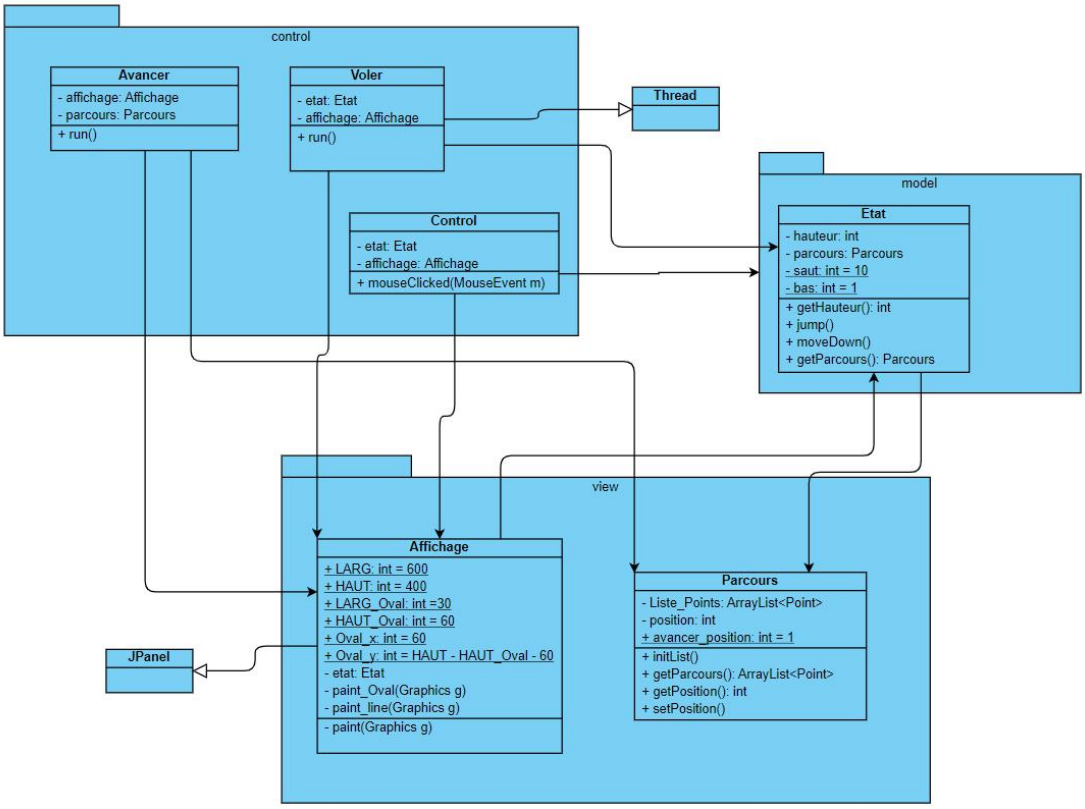
Pour initialiser la liste de points:
Constructeur: Parcours()

```
    Liste_Points = new ArrayList<Point>()
    int x <-    Affichage.Oval_x + Affichage.LARG_Oval/2;
    int y <- Affichage.Oval_y + Affichage.HAUT_Oval/2;
    Liste_Points.add( new Point(x,y) );
    initList();

initList()
    int x <- Liste_Points[length].x;
    int y;
     Faire
          x <- x + ( new Random( ).nextInt( Affichage.LARG));
          y<-y+(new Random().nextInt(Affichage.HAUT - 2*Affichage.HAUT_Oval)
               + Affichage.HAUT_Oval );
          Liste_Points.add( new Point(x,y) );
     Tant que x < Affichage.LARG
```



2.3 Pour défilement automatique de la ligne brisée, on ajoute une variable position dans la classe Parcours et crée une classe Avancer qui implémente un thread pour faire avancer la position. En fin, on motifie la classe Parcours pour que la liste des points caractéristiques s'adapte à la position du joueur sur le parcours.



3.1 Pour trouver la position ordonnée de la ligne brisée correspondant à la position en abscisse du centre de l'ovale. Tout d'abord, nous parcourons la liste des points et comparons l'abscisse du centre de l'ovale x avec les abscisses des éléments de la liste, afin de trouver les deux points (x1, y1), (x2, y2) sur les côtés gauche et droit de l'ovale. Ensuite, nous avons calculé l'équation de la droite passant par ces deux points. Enfin, ramenez l'abscisse du centre de l'ovale dans cette formule pour obtenir l'ordonnée correspondant à la ligne brisée y_line.

Utilisez deux points pour représenter l'équation d'une ligne droite les traversant:

(x - x1) / (x2 - x1) = (y - y1) / (y2 - y1)

Donc

(x - x1) / (x2 - x1) = (y_line- y1) / (y2 - y1)

y_line = (x - x1) * (y2 - y1) / (x2 - x1) + y1

Ou soit l'équation de la droite:

y = kx + b

k = (y2 - y1) / (x2 -x1)

b = (x2*y1 - x1*y2) / (x2 - x1)

y_line = kx +b

Pour détecter les collisions. En utilisant cette formule, écrivez une fonction testPerdu dans la classe Etat qui renvoie vrai si l'ovale est sorti de la ligne brisée.

Les conditions de la détection de l'ovale n'est pas sorti de la ligne brisée sont:

y <= y_line et y_line <= (y + Affichage.HAUT_Oval)

Après avoir détecté l'ovale est sorti de la ligne brisée:

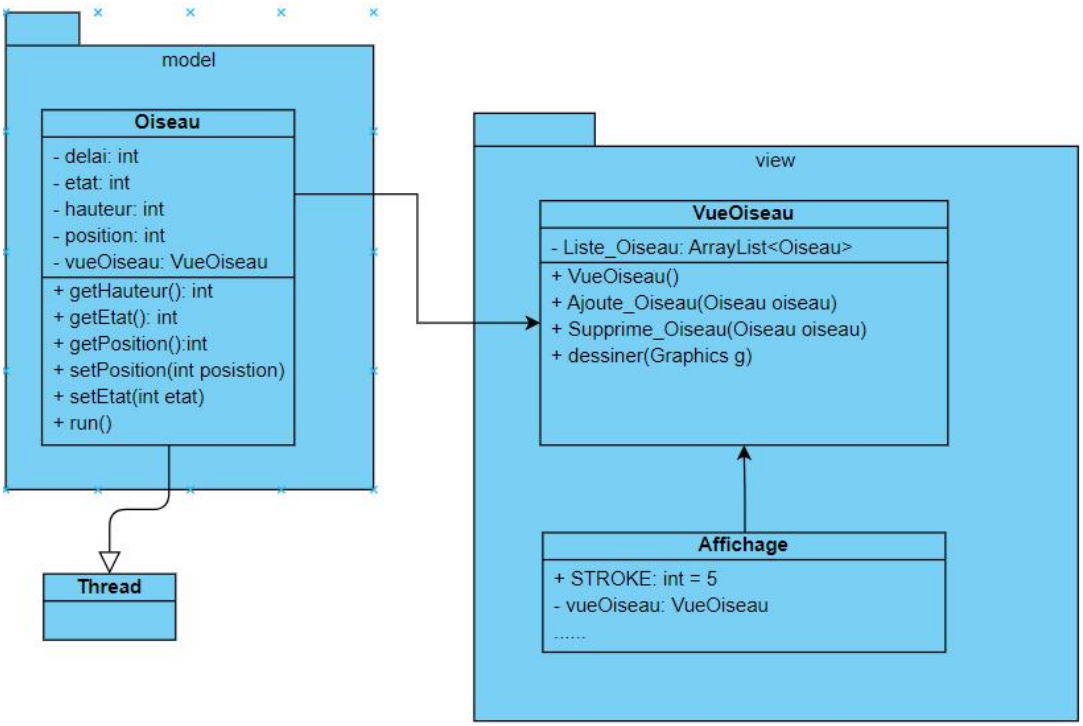
Pour s'arrêter les threads de vol et d'avancement du parcours, nous avons modifié la condition dans les while().

Pour faire de L ' interface ne réagisse plus aux clics souris, Nous utilisons removeMouseListener pour supprimer MouseListener.

Pour afficher le score de l'utilisateur, nous utilisons la classe JOptionPane pour faire apparaître une nouvelle fenêtre quand le thread avancer s'arrête.

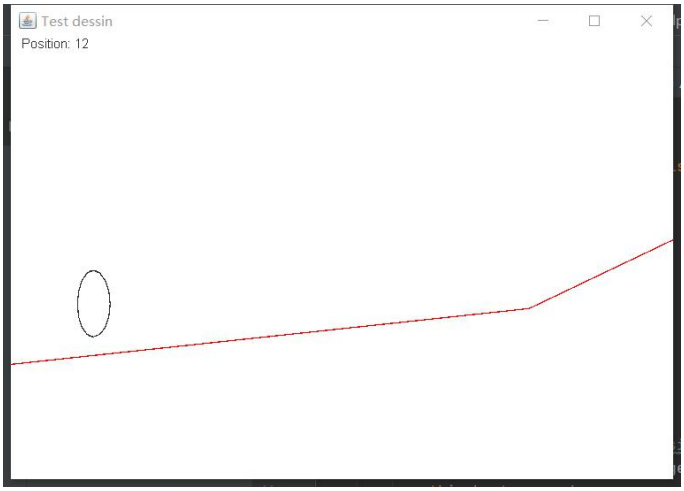
4.1 Pour améliorer des dessins, on utilise les Stroke pour dessiner un ovale et une ligne brisée plus épais.

4.2Pour ajouter d'éléments de décors, on crée une classe Oiseau qui définie les données de l'oiseau et hérite de Thread, La méthode run de ce thread utilisera une boucle pour motifié les données de l'oiseau. Ensuite on on crée une classe VueOiseau qui possède un attribut de type ArrayList<Oiseau> et qui possède une méthode dessiner(Graphics g) pour gérer l'affichage des oiseaux.

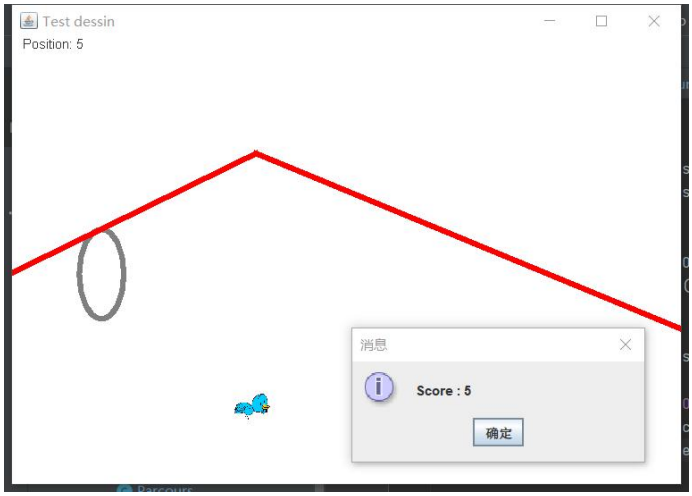


Résultat

Dans la deuxième partie du projet (séance 2)



Dans la troisième partie du projet (séance 3)



Documentation utilisateur

Prérequis : 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 exécutable) : double-cliquez sur l'icône du fichier .jar. Cliquez sur la fenêtre pour faire monter l'ovale.

Documentation développeur

La classe qui contient la méthode main est: La classe Main

Les principales constantes pouvent etre modifié pour changer le fonctionnement du code sont:

- "SAUT"; "BAS": dans la classe Etat
- "AVANCER_POSITION": dans la classe Parcours

Séance 1：

Les prochaines fonctionnalités seront la création d'une ligne brisée et son déplacement automatique pour donner l'impression que l'ovale avance le long de cette ligne.

Séance 2：

Le prochaine fonctionnalité est la réalisation de la détection des collisions.

Séance 3:

Nous avons terminé la réalisation de toutes les principales fonctionnalités. Et nous avons fait quelques améliorations..

Conclusion et perspectives

Dans la première partie du projet (séance 1),

Nous avons réalisé Ces deux sous-fonctionnalités

- création d'une fenêtre dans laquelle est dessiné l'ovale ;
- déplacement de l'ovale vers le haut lorsqu'on clique dans l'interface.

Et, nous avons adopté le motif MVC pour le développement de notre interface graphique.

Notre prochaine tâche est de réaliser les fonctionnalitésde la ligne brisée.

Dans la deuxième partie du projet (séance 2),

nous avons réalisé ce sous-ensemble de fonctionnalités:

- l'ovale perde de l'altitude lorsque l'utilisateur ne clique pas sur l'écran;
- Dessiner la ligne brisée dans la fenêtre
- défilement automatique de la ligne brisée

Notre prochaine tâche est de réaliser la détection des collisions.

Dans la troisième partie du projet (séance 3),
Nous avoir fait un système fonctionnel est de détecter les collisions.
Ensuite, nous avons fait deux améliorations supplémentaires:

- Amélioration des dessins
- Ajout d'éléments de décors