Rapport de projet PCII

Ovale en mouvement

L3 Informatique

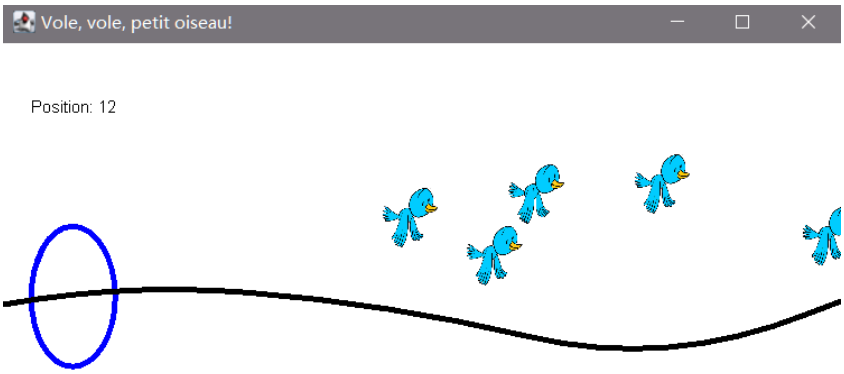
Zhenhai XU

Groupe : 2

Sommaire

INTRODUCTION

Nous voulons créer un mini-jeu inspiré des oiseaux qui volent. Il y a une courbe définie. Nous contrôlerons une ovale. L'ovale avancera et tombera automatiquement avec le temps. Nous tapons sur l'écran pour faire monter et déplacer l'ovale. Faites bouger l'ovale le long de la courbe. Si l'ovale quitte la ligne brisée, le jeu se terminera. Voici à quoi pourrait ressembler l'interface graphique de notre jeu :



ANALYSE GLOBALE

Il y a quatre fonctionnalités principales : l'interface graphique avec l'ovale et la ligne brisée, le défilement automatique de la ligne brisée, la réaction de l'ovale aux clics de l'utilisateur, Opération en fin de partie, Je les ai divisés en parties suivantes :

Premier partie :

I. Création d'une fenêtre dans laquelle est dessiné l'ovale ;
II. Lorsque on clique dans l'interface, faites monter l'ovale vers le haut.

Deuxième partie :

I. Tracez une ligne brisée dans la fenêtre.
II. Déplacez la ligne brisée vers la gauche.
III. Laissez le segment de ligne brisée s'étendre indéfiniment.

Troisième partie :

I. Conception de la formule de collision de ligne et de cercle.
II. Les threads de vol et d'avancement du parcours s'arrêtent.
III. Arrêter la surveillance de la souris.
IV. Un message s'affiche avec le score de l'utilisateur.

Quantième partie :

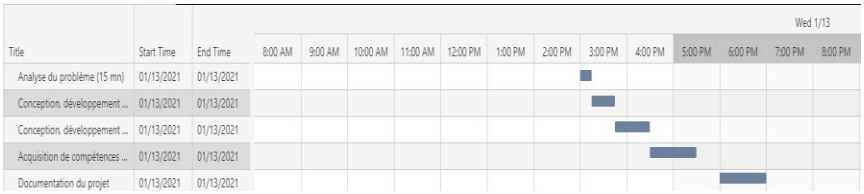
I. Transforme la ligne brisée en courbe.
II. Ajouter plusieurs oiseaux animés.
III. Construire une archive Java (fichier .jar) exécutable.

PLAN DE DEVELOPPEMENT

Liste des tâches :

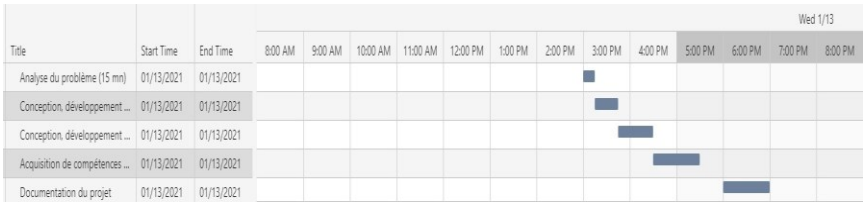
Première partie :

I.	Analyse du problème.	(15 mn)
II.	Conception, développement et test d'une fenêtre avec un ovale.	(30 mn)
III.	Conception, développement et test du mécanisme de déplacement de l'ovale.	(45 mn)
IV.	Acquisition de compétences en Swing.	(60 mn)
V.	Documentation du projet.	(60 mn)



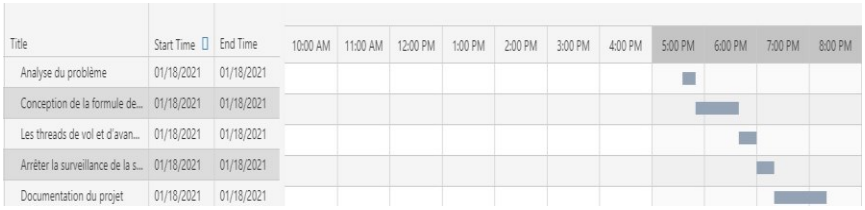
Deuxième partie :

I.	Analyse du problème.	(15 mn)
II.	Conception, développement et test de générer une ligne brisée.	(30 mn)
III.	Conception, développement et test du mécanisme de déplacement de la ligne brisée.	(45 mn)
IV.	Documentation du projet.	(60 mn)



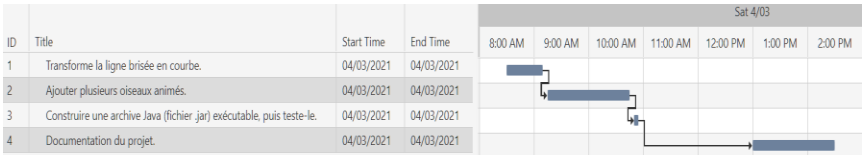
Troisième partie :

I. Analyse du problème.	(15 mn)
II. Conception, développement et test de collision de ligne et de cercle.	(50 mn)
III. Conception arrêter la surveillance de la souris.	(20 mn)
IV. Un message s'affiche avec le score de l'utilisateur.	(20 mn)
V. Documentation du projet.	(60 mn)



Quantième partie :

I. Transforme la ligne brisée en courbe.	40 mn
II. Ajouter plusieurs oiseaux animés.	90 mn
III. Construire une archive Java (fichier .jar) exécutable, puis teste-le.	5 mn
IV. Documentation du projet.	90 mn



CONCEPTION GENERALE

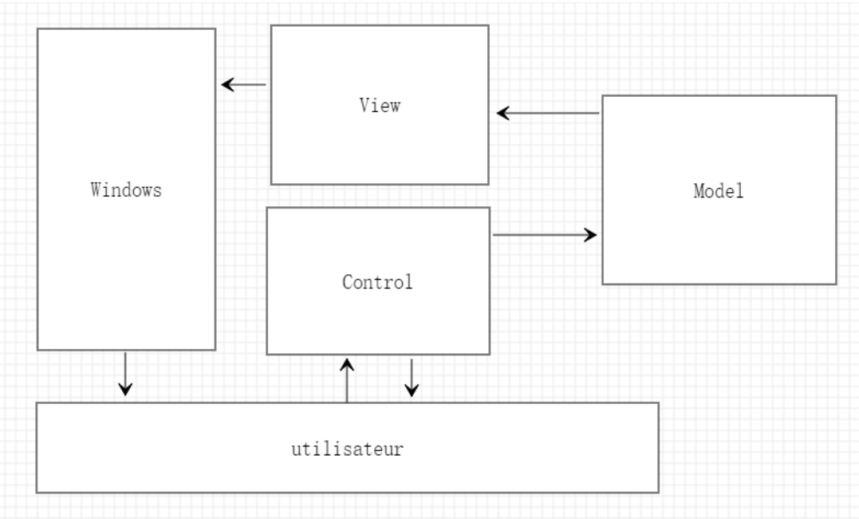
L'architecture de développement est basée sur le motif "Model Vue Contrôle" (MVC). Ce modèle nous permet de bien organiser le code source, cela nous permet de définir clairement le rôle de chaque fichier. J'ai divisé la source du code en trois parties.

Model : Le modèle contient les classes d'objets que nous dessinons sur l'interface, les oiseaux, l'ovale et les courbes.

Vue : Les données acquises sont dessinées sur l'interface.

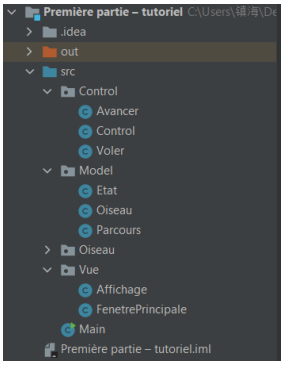
Control : Contient les mécanismes du jeu, utilise des Thread pour démarrer le jeu, surveille les actions du joueur.

La figure suivante est un diagramme schématique du Framework MVC：

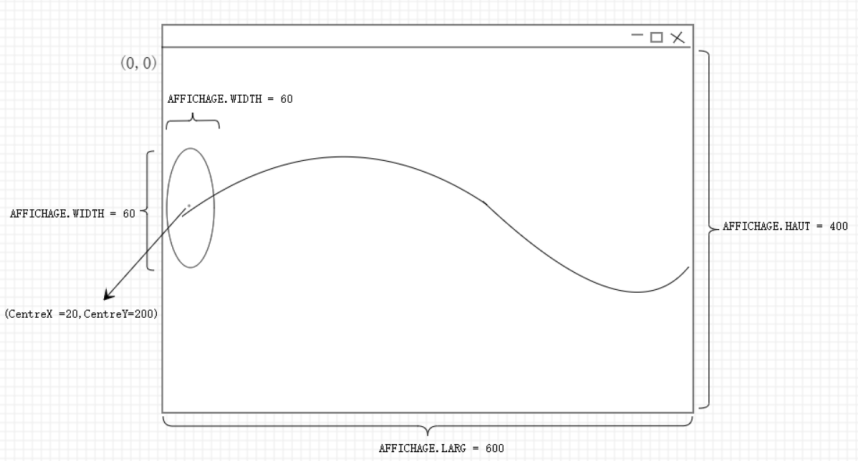


Je sépare **Main.java** des trois dossiers MVC, et le dossier Oiseau contient les images des oiseaux que nous avons chargés.

Ce qui suit est un diagramme schématique du dossier de code:



L'interface graphique est comme indiqué ci-dessous



La taille de la fenêtre et la taille de l'ovale sont définies dans **affichage.java**. Les coordonnées de l'interface (0,0) sont situées dans le coin supérieur gauche de la fenêtre. Cliquer sur l'écran fera monter l'ovale, sinon l'ovale restera chute.

CONCEPTION DETAILLEE

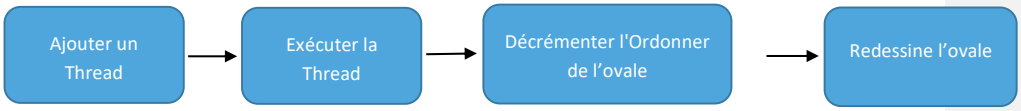
I. CONCEPTION D'UNE FENETRE AVEC UN OVALE.

Classe utilisé : Affichage.java

Dans **affichage.java** on définit la taille de la fenêtre et la taille de l'ovale.
Affichage.java hérite de JPanel, on utilise la méthode **g.drawOval ()** pour dessiner l'ovale.

II. CONCEPTION DU MECANISME DE DEPLACEMENT DE L'OVALE.

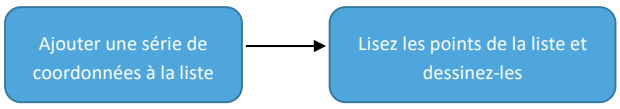
Classe utilisé : Voler.java Affichage.java



Ajoutez un thread. Lorsque le thread est en cours d'exécution, augmentez l'ordonnée de l'ellipse à chaque fois, puis redessinez l'ovale pour que l'ovale semble se déplacer vers le bas.

III. CONCEPTION DE GENERER UNE LIGNE BRISEE.

Classe utilisé : Affichage.java parcours.java



Ajoutez plusieurs coordonnées à la liste, lisez deux points de coordonnées (le premier point et le deuxième point, le deuxième point et le troisième point, etc.) à la fois via la boucle for, puis utilisez **Affichage.drawline()** pour dessiner une ligne brisée connectée.

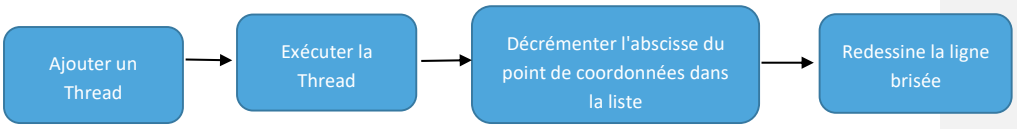
批注 [WU1]:

批注 [WU2]:

批注 [WU3]:

IV. CONCEPTION DU MECANISME DE DEPLACEMENT DE LA LIGNE BRISEE.

Classe utilisé : Affichage.java Avance.java Etat.java



Créez un thread à exécuter en continu, chaque exécution du thread décrémentera automatiquement l'abscisse du point de coordonnées dans la liste, puis redessine une nouvelle ligne brisée via **Affichage.repaint()**, ce qui donne l'impression que le segment de ligne se déplace vers la gauche.

Le point de coordonnées à l'extrémité gauche de l'écran sera supprimé.

V. CONCEPTION COLLISION DE LIGNE ET DE CERCLE.

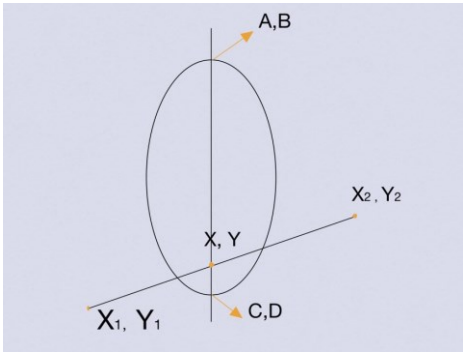
Afin d'effectuer le test de collision, nous devons utiliser des formules mathématiques :

$$\frac{x - x_1}{x_2 - x_1} = \frac{y - y_1}{y_2 - y_1}$$

Grâce à la formule ci-dessus, nous pouvons obtenir :

$$y = \frac{x - x_1}{x_2 - x_1} \times (y_2 - y_1) + y_1$$

Nous savons que la coordonnée sur l'axe x du point central du cercle est X et les coordonnées de la ligne brisée testé sont également connues (x1, y1 et x2, y2). Grâce à cette formule, nous pouvons trouver le Y dans cette ligne brisée. Si Y est en dehors de l'ovale (En d'autres termes, Y n'est pas entre B et D), cela signifie que la ligne segment et l'ovale ne se croisent pas, c'est-à-dire que notre ovale est hors ligne brisée, alors le jeu est terminé.



Après avoir augmenté l'épaisseur, nous devons considérer que la longueur verticale de la ligne diagonale augmentera davantage. Grâce à la fonction trigonométrique, nous pouvons calculer l'augmentation de la longueur de la ligne diagonale, puis effectuer une détection de collision avec l'ellipse épaissie.

Classe utilisé : Etat.java

Le code est implémenté comme suit :

```
k est la pente
b est le montant de la translation de la ligne brisee.
a est l'angle de la ligne horizontale après l'inclinaison de la ligne breisee
*/
double line = 0;double k = 0;double b = 0;

ArrayList<Point> pa = parcours.ajoutePoint();
/*Trouvez les deux points actuels de la ligne breisee*/
for(int i=0 ; i < pa.size()-1;i++) {
    if (x >= pa.get(i).x && x <= pa.get(i + 1).x) {
        x1 = pa.get(i).x;
        x2 = pa.get(i + 1).x;
        y1 = pa.get(i).y;
        y2 = pa.get(i + 1).y;
    }
}
//Appliquer des formules mathématiques,Calculer les valeurs de k et b
k = (y2 - y1)/(x2 - x1);
b = (x2*y1 - x1*y2)/(x2 - x1);
//Calculer la valeur de y
line=k*x+b;

//Si la ligne brisee entre en collision avec l'ovale, le joueur échoue.
if( line < this.hauteur + h  || line   > this.hauteur+Affichage.HEIGHT -  h)
    return true;
else
    return false;
```

Avant de calculer leur point d'intersection, nous devons déterminer sur quel segment de ligne se trouvent les sommets supérieur et inférieur de l'ellipse, puis calculer la valeur y via la formule, puis nous pouvons déterminer si le jeu est terminé.

VI. CONCEPTION D'ARRETER LA SURVEILLANCE DE LA SOURIS.

Classe utilisé : Affichage.java Control.java

Utilisez **MouseListener** dans **Control.java** et obtenez les actions du joueur en ajoutant **MouseListener** à l'écran. Si le joueur clique sur l'écran, nous utiliserons la méthode **mouseClicked ()** pour modifier l'ordonnée de l'ellipse, puis utiliserons la méthode **Affichage.repaint ()** pour redessiner l'ovale afin que l'ovale semble se déplacer vers le haut.

VII. TRANSFORME LA LIGNE BRISEE EN COURBE

Classe utilisé : Affichage.java parcours.java

Grâce à **QuadCurve2D** dans la bibliothèque, nous pouvons utiliser les points de coordonnées de la liste pour générer la courbe. La courbe de Bézier a besoin de trois points pour dessiner la courbe, nous utilisons donc la boucle FOR pour dessiner la courbe en utilisant trois points à chaque fois.

```
for (int i = 0; i < L1.size() - 2; i++) {
    double p0x = (L1.get(i).getX() + L1.get(i + 1).getX()) / 2;
    double p0y = (L1.get(i).getY() + L1.get(i + 1).getY()) / 2;
    double p1x = (L1.get(i + 1).getX() + L1.get(i + 2).getX()) / 2;
    double p1y = (L1.get(i + 1).getY() + L1.get(i + 2).getY()) / 2;
    Point2D.Double start = new Point2D.Double(p0x , p0y);
    Point2D.Double ctrl = new Point2D.Double(L1.get(i + 1).getX(), L1.get(i + 1).getY());
    Point2D.Double fin = new Point2D.Double(p1x , p1y);
    courbe.setCurve(start, ctrl, fin);
    g.draw(courbe);
```

VIII. AJOUTER PLUSIEURS OISEAUX ANIMES.

Classe utilisé : Oiseau.java Avancer.java Affichage.java

Oiseau a hérité du Thread, utilisez la méthode **run()** dans Oiseau.Java, de sorte que l'oiseau charge continuellement de nouvelles images sous le fil du fil, faisant ressembler l'oiseau à une animation. Générer un nouvel oiseau dans un Thread **Avancer.java**, et faire bouger chaque oiseau vers la gauche, et l'oiseau qui sort de l'écran sera supprimé.

IX. L'INTERFACE DE FIN DE JEU

Classe utilisée : Voler.java Control.java Avancer.java

```
public void run() {
    while (!etat.perdu) {
        try {
```

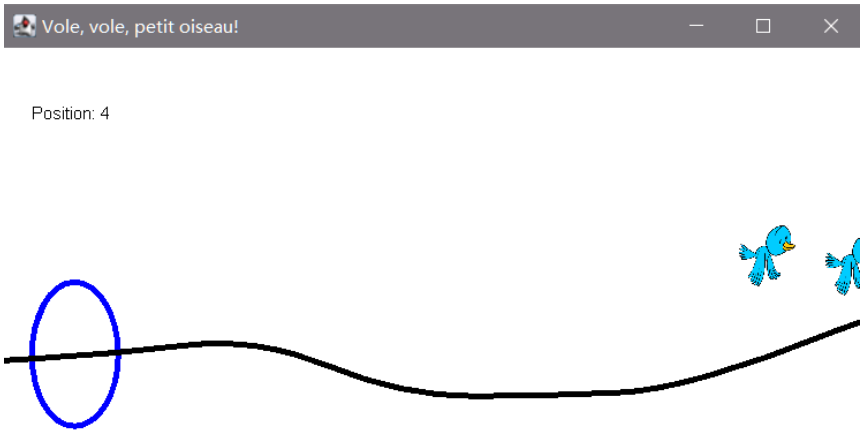
```
@Override
public void run() {
    while (!etat.perdu) {
        try {
            Thread.sleep(200);
            etat.moveDown();
            etat.perdu=etat.testPerdu();
            parcours.removeParcours();
            affichage.revalidate();
            affichage.repaint();

        } catch (InterruptedException e) {
            e.printStackTrace();
        }
    }
    JOptionPane.showMessageDialog(null, "votre score est de : " + etat.getParcours().getPosition() );
}
```

```
public void mouseClicked(MouseEvent e) {
    //si joueur a perdu, l'interface ne réagisse
    if (etat.testPerdu()){
        affichage.removeMouseListener(this);
```

À la fin du jeu, nous arrêtons la **MouseListener**, arrêtons les Threads et affichons le score.

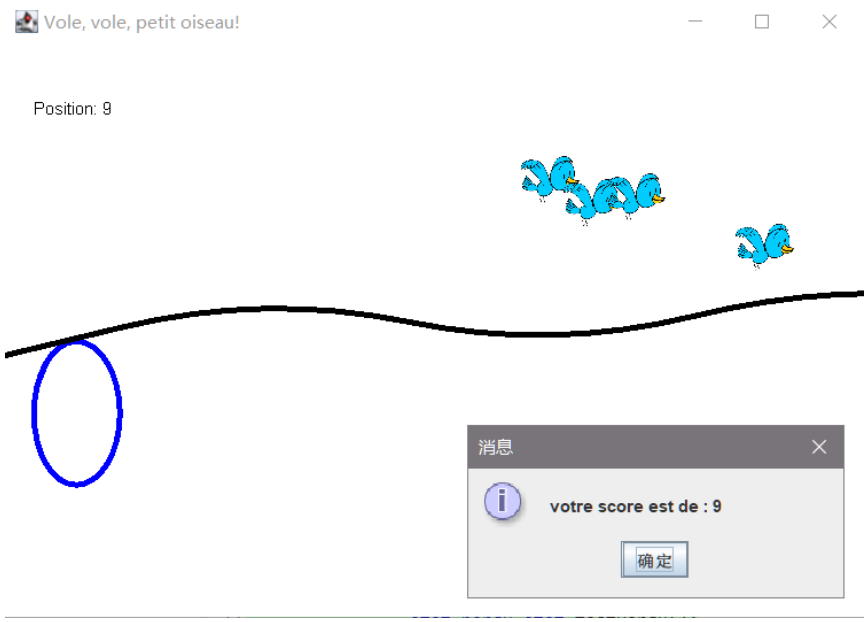
RESULTAT

ÉCRAN DE DEMARRAGE DU JEU



Le jeu commence. La position de l'ellipse est à gauche de l'écran et l'oiseau apparaît à droite.

L'ECRAN DE FIN



L'ellipse frappe la courbe et le jeu est terminé.

Les Threads s'est arrêté et une boîte de dialogue indiquant le score est apparue.

DOCUMENTATION UTILISATEUR

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.

Mode de jouer : gardez l'oiseau dans la courbe en clique l'écran, et le jeu se terminera et vous invitera à obtenir le score

DOCUMENTATION DEVELOPPEUR

Ce jeu utilise le modèle MVC, on trouve très bien la fonction correspondante. Dans quel dossier se trouve java. Si vous souhaitez modifier l'ellipse et la courbe, vous pouvez trouver la méthode de dessin de l'ellipse et de la courbe dans **Affichage.java** Modifiez la photo de l'oiseau uniquement dans oiseau. Pour modifier le chemin de l'image en **Oiseau.java**.

Les fonctions que nous prévoyons d'ajouter :

Rendre l''écran plus riche, plus de décorations.

Avoir un mécanisme de récompense.

Le jouer pouvez choisir la difficulté et la vitesse pour rendre le jeu plus difficile.

CONCLUSION

Nous avons complété toutes les fonctions de base de ce projet, la courbe en mouvement, l'ellipse contrôlée et le mécanisme pour juger de la fin du jeu. De plus, j'ai ajouté certain nombre d'animations d'oiseaux comme décorations, transformant la ligne brisée en courbe. J'espère qu'à l'avenir, ce jeu pourra être rendu plus stimulant, avec de meilleurs graphismes, une musique et une interface de démarrage pourront être ajoutées.

Mais dans ce projet, j'ai aussi rencontré des difficultés, la première difficulté rencontrée dans cette recherche a été l'animation des oiseaux. Après avoir essayé, j'ai très bien résolu le problème, puis j'ai généré plusieurs oiseaux. Cela m'a fait réfléchir longtemps, et finalement je l'ai fait aussi.

En ce qui concerne l'utilisation de GitHub, en raison du manque de bonne compréhension du fonctionnement de ce site Web, le travail de téléchargement du projet est entravé. Et comme le chemin de GitHub est différent du chemin où je stocke le logiciel sur mon ordinateur, après que quelqu'un ait téléchargé mon code sur GitHub, il n'y a aucun moyen de charger l'image de l'oiseau lors de l'exécution et une erreur se produit, mais j'ai trouvé le problème et l'a résolu. Le rapport est la partie la plus difficile de ma tâche pour ce projet, mais après avoir terminé le rapport, j'ai constaté que je comprenais mieux mon projet et que je pouvais le présenter aux autres de manière bien réglementée. Je pense qu'en travaillant plus tard, J'ai souvent besoin d'une équipe de coopération, un bon rapport est très important, cela peut aider les autres à comprendre mon travail plus rapidement, je pense avoir beaucoup appris.