Université Paris Saclay

Faculté des Sciences

PROJET de PCII

Ketchupp Circle

Présenté par :

Pourchot Alistair

Groupe : LDD3-MAG-INFO

Introduction

Nous voulons réaliser un jeu inspiré de Ketchapp Circle, un jeu dans lequel un ovale se déplace le long d'une ligne brisée générée aléatoirement. Le but de ce jeu est d'éviter de sortir de cette ligne. Le joueur peut faire sauter l'ovale, qui va ensuite redescendre par lui-même.

Contents

1 Analyse

1.1 Analyse globale

Cette analyse présente les différentes fonctionnalités implémentées dans le jeu. Afin de structurer la développement et conserver une architecture modulaire, les fonctionnalités du projet on été découpées selon 5 pôles majeurs. Cette approche nous permet de séparer clairement la logique physique, la gestion de l'environnement et l'interface utilisateur.

1. Mécaniques du jeu
2. Environnement et génération
3. Interface utilisateur
4. Cycle de vie du jeu
5. Esthétique

1.2 Analyse détaillée

On détaille ici un peu plus les fonctionnalités que l'on a ajouté.

- Fonctionnalité 1 : (Mécaniques)
  - Pouvoir faire monter l'ovale
    - > Difficulté : Modérée
    - > Priorité : 1
  - Faire monter l'ovale lorsque le joueur appuie sur *espace* ou lorsqu'il *clique*
    - > Difficulté : Facile
    - > Priorité : 1
  - Faire descendre l'ovale lorsqu'on ne clique pas
    - > Difficulté : Facile
    - > Priorité : 1
  - Utiliser une gravité pour créer un mouvement de saut
    - > Difficulté : Modérée
    - > Priorité : 2

  - Perte de vie lorsque le joueur sort de la ligne
    > Difficulté : Difficile
    > Priorité : 1

- Fonctionnalité 2 : (Environnement)
  - Génération d'une ligne brisée aléatoire
    > Difficulté : Modérée
    > Priorité : 1
  - Ajout et retrait de points à la ligne pour simuler une ligne infinie
    > Difficulté : Modérée
    > Priorité : 1
  - Scroll de la ligne brisée
    > Difficulté : Modérée
    > Priorité : 1
  - Accélérer la vitesse de l'avancement à partir d'un certain score
    > Difficulté : Facile
    > Priorité : 3

- Fonctionnalité 3 : (Interface)
  - Dessiner un ovale dans une fenêtre graphique
    > Difficulté : Facile
    > Priorité : 1
  - Dessiner un sol vert et un fond bleu
    > Difficulté : Facile
    > Priorité : 3
  - Dessiner une barre de vie
    > Difficulté : Facile
    > Priorité : 3
  - Ambiance rougeâtre lorsqu'on perd beaucoup de vie
    > Difficulté : Modérée
    > Priorité : 3
  - Affichage du score du joueur
    > Difficulté : Facile

> Priorité : 2

- Fonctionnalité 4 : (Vie du jeu)
  - Ajout d'un menu du jeu
    > Difficulté : Difficile
    > Priorité : 2
  - Ajout d'un menu de fin de jeu
    > Difficulté : Modérée
    > Priorité : 2
  - Fin de jeu lorsque la vie atteint 0
    > Difficulté : Facile
    > Priorité : 1
  - Réinitialisation de toutes les variables pour nouvelle partie
    > Difficulté : Facile
    > Priorité : 1
- Fonctionnalité 5 : (Esthétique)
- Ajout de nuages
  > Difficulté : Modérée
  > Priorité : 3
- Ajout d'oiseaux indépendants
  > Difficulté : Difficile
  > Priorité : 3

2 Plan détaillé

- Apprentissage de Swing : 1h
- Apprentissage des Threads : 30min
- Dessin de l'ovale : 15min
- Événement souris et clavier : 20min
- Thread de mouvement : 20min

- Ajout gravité : 20min
- Rédaction du document d'analyse et conception : 8h
- Dessin ligne brisée aléatoire : 1h
- Scroll : 1h
- Ajout de points à la ligne brisée : 30min
- Arrêt du jeu : 2h
- Barre de vie : 30min
- Score : 30min
- Nuages : 30min
- Oiseaux : 1h30
- Décor : 1h
- Stress par perte de vie : 30min
- Ajout de boutons : 30min
- Re-factorisation du code : 1h30
- Ajout commentaires : 30min

2.1 Diagramme de Gantt

Pour représenter la répartition du travail de manière visuelle, voici un diagramme de Gantt.

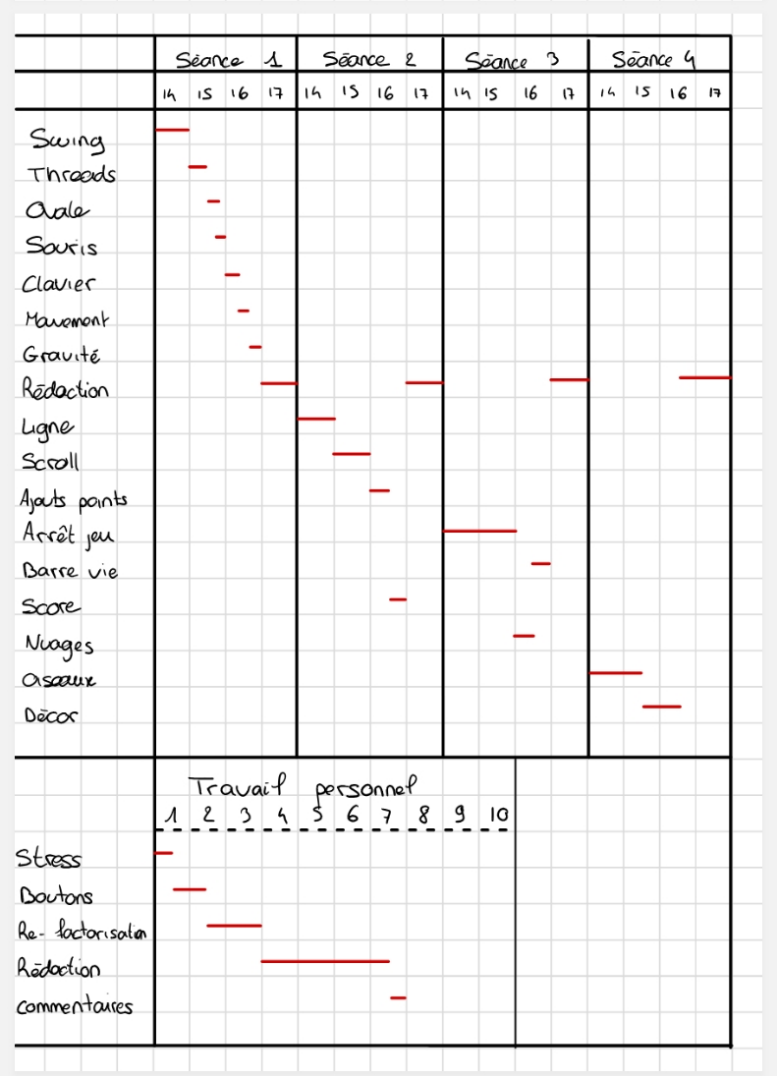

Figure 1: Diagramme de Gantt

3 Conception

3.1 Conception générale

Nous avons décidé de suivre le procédé Modèle Vue Contrôleur (MVC) pour la gestion de ce projet. Nous avons donc :

- Dans le Modèle :
  - vitesse, hauteur, scroll, saut

- Dans la Vue : on retrouve tout ce qui concerne l'affichage
  - ovale, ligne, décor
- Dans le Contrôleur
  - Interactions avec la souris et le clavier, thread de chute / montée et repaint

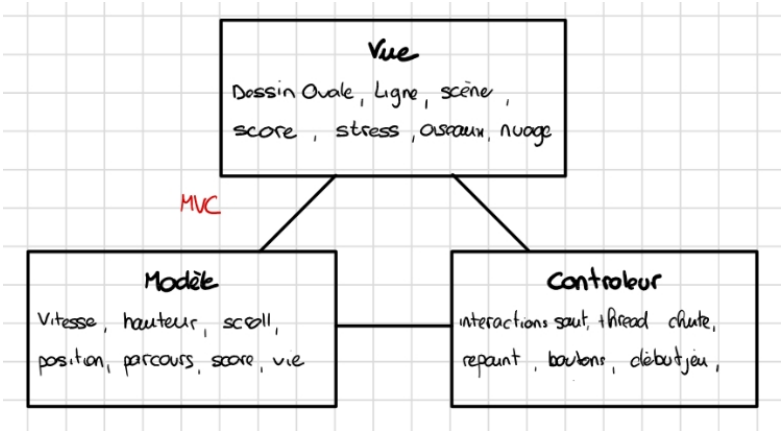

Figure 2: Patron du modèle MVC

3.2 Conception Détaillée

Dessin ovale

Utilisation de l'API Swing et de la classe JPanel.
Définition des dimensions de l'ovale et de la fenêtre $HAUTEUR$, $LARGEUR$, $ZONE\_HAUT$, $ZONE\_LARG$
Voici un diagramme de classe simplifié :

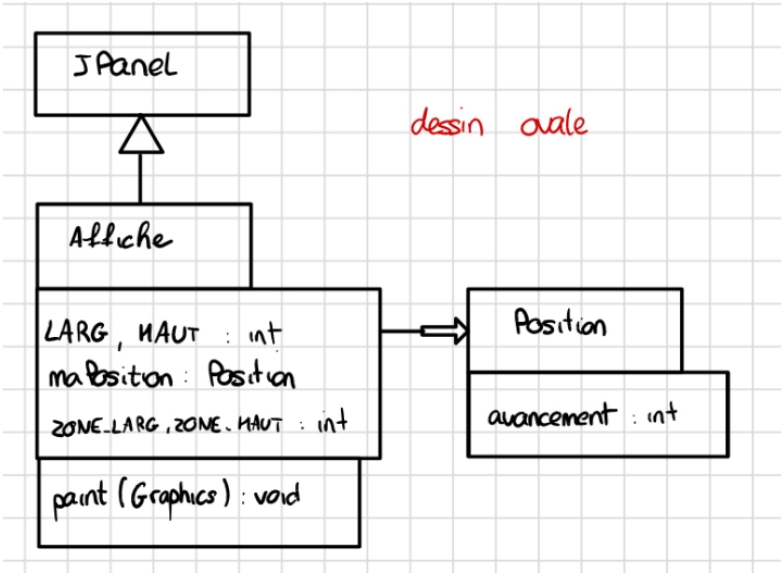

Figure 3: Diagramme dessin de l'ovale

Dessin ligne brisée

Définition de constantes $largeur$, $position$, $fenetre$
Voici un diagramme de classe simplifié :

Variables $ECART\_MIN$ et $ECART\_MAX$ concernant les conditions d'écart horizontaux entre les différents points. On s'occupe de la génération des points à l'aide d'un algo que l'on appelle dans le constructeur après avoir créé nos deux premiers points manuellement pour faire une ligne droite.
Algo :

```
Entrée : un point Précédent
Sortie : un point Nouveau

ECART_X = ECART_MIN + RANDOM(ECART_MAX - ECART_MIN +1);
X = PRECEDENT.X + ECART_X
Y = ECART_Y_MIN + RANDOM(HAUTEUR_MAX)
return Point(X, Y)
```

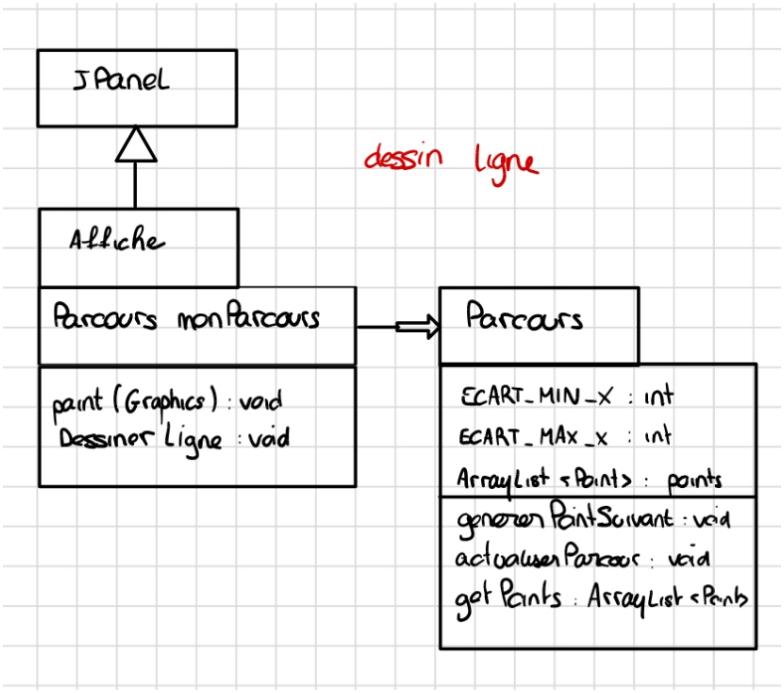

Figure 4: Diagramme ligne brisée

Programmation événementielle

On s'occupe ici des interactions entre le joueur et la machine. On utilise la classe MouseListener pour les interactions avec le clic gauche pour sauter, et la classe KeyListener pour les interactions avec la barre espace (pour sauter aussi). La hauteur du saut est définie dans une constante *hauteurSaut*. Le thread de mise à jour de l'affichage a un délai de 50ms

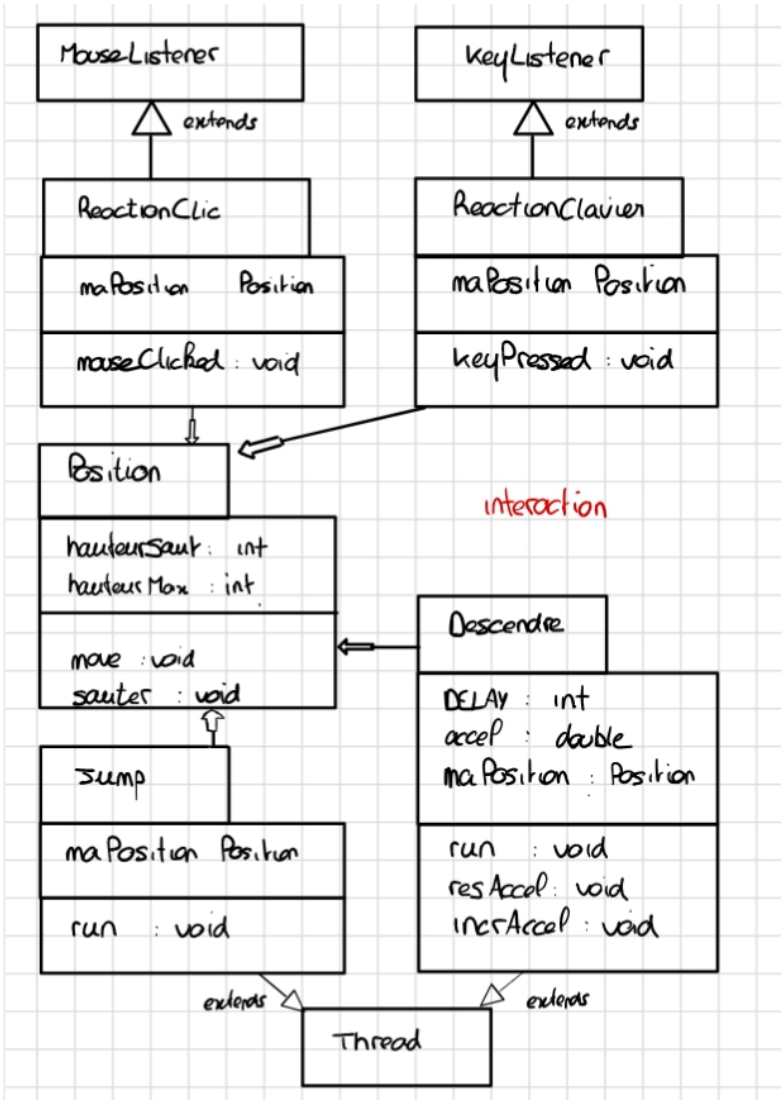

Figure 5: Diagramme interaction joueur

Début et Arrêt de jeu

On rajoute une quantité de vie dans une variable *nb_vies* que l'on décrémente à chaque instant ou le cercle est en dehors de la ligne. Une fois que le jeu est fini, le joueur a la possibilité de cliquer sur un bouton pour rejouer. Dans ce cas, on dispose d'une fonction reset qui nous permet de réinitialiser nos parcours, position, etc.

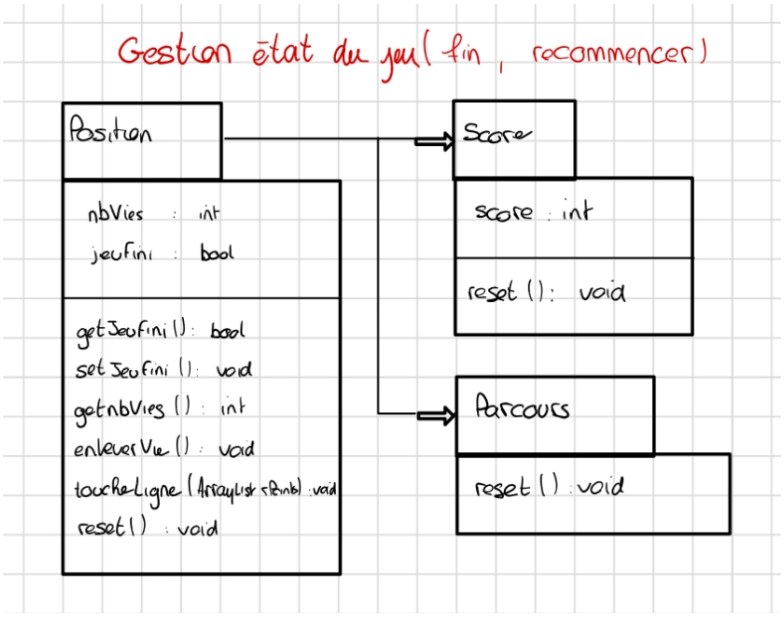

Figure 6: Diagramme gestion état du jeu

Interactions boutons

Pour gérer les interactions avec les boutons, on dispose d'une autre classe dans le contrôleur qui gère les différents traitements à faire selon les boutons.

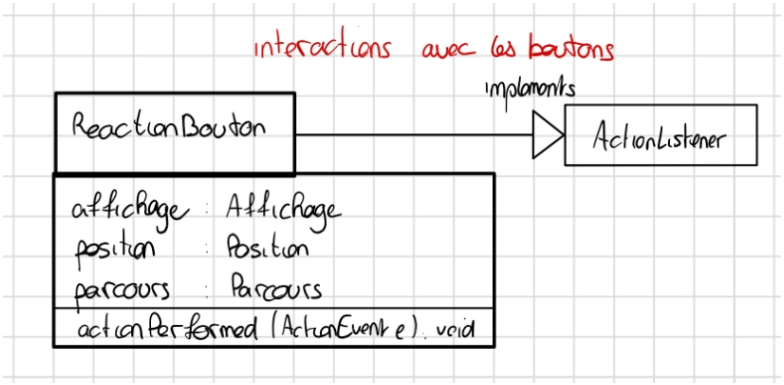

Figure 7: Diagramme boutons

Score

On dispose d'une classe score, que l'on utilise pour incrémenter notre score à chaque génération d'un nouveau point. Cette classe ne sert qu'à initialiser *score* et l'incrémenter.

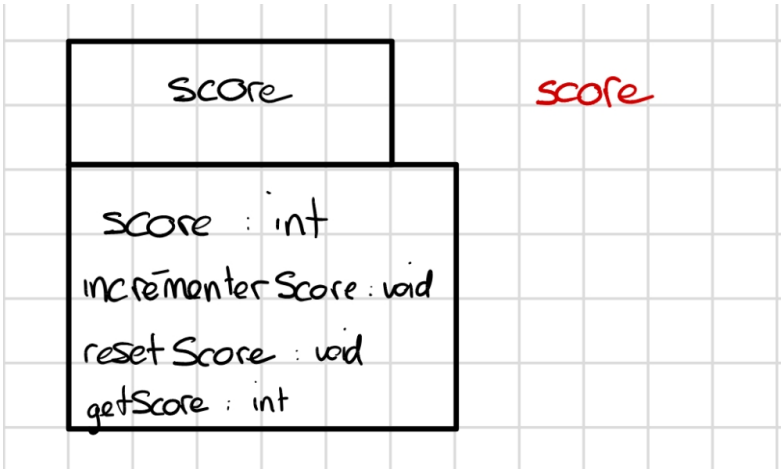

Figure 8: Diagramme du score

Oiseaux

On a une classe Oiseaux qui, lorsqu'on l'appelle, créer un certain nombre d'oiseaux (indépendants) avec leur propre thread et une vitesse *delay* aléatoire initialisé dans une sous-classe Oiseau. La position des différents oiseaux se fait grâce à des attributs x, y initialisés dans la classe.

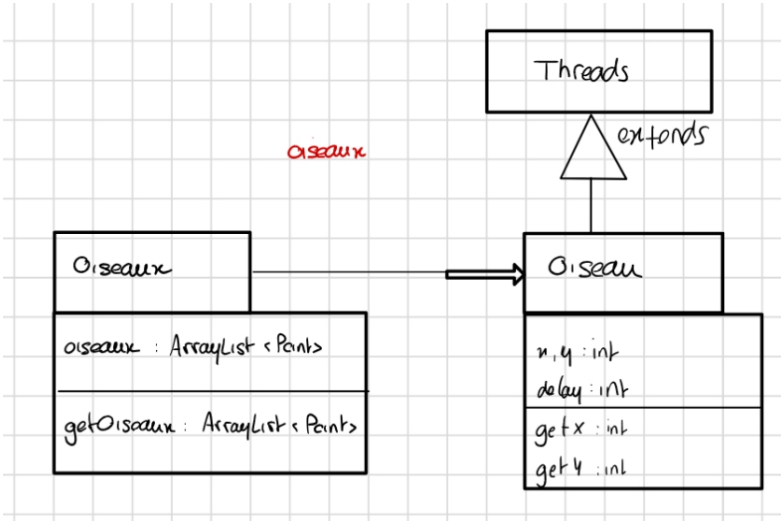

Figure 9: Diagramme oiseaux

4 Aperçu du résultat

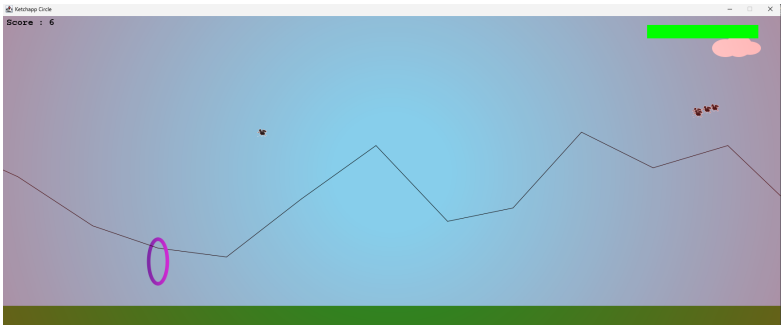

Figure 10: Rendu du jeu

On peut distinguer sur cette image : les nuages, les oiseaux, la barre de vie, le score, la ligne brisée, le cercle et le décor.

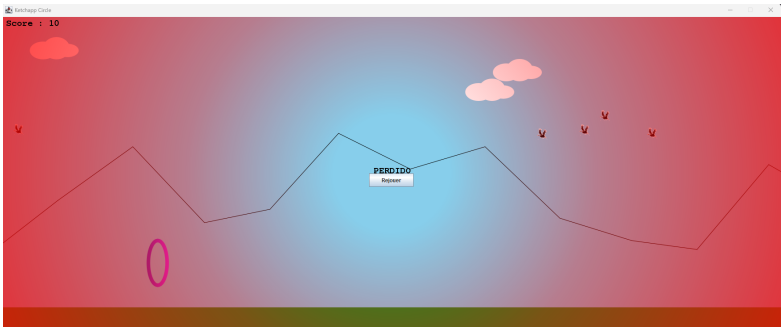

Figure 11: Fin de partie

On remarque bien que lorsque l'on perd de la vie, une sorte d'aura rouge se met sur les bords de notre écran pour montrer au joueur qu'il est dans une situation de plus en plus critique.

Figure 12: Évolution de la vie

Notre barre de vie suit un dégradé allant du vert au rouge.

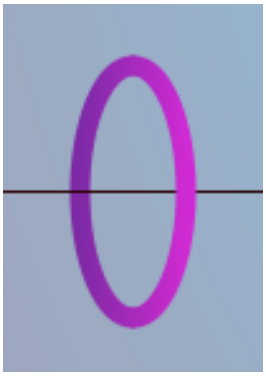

Figure 13: ovale

Nous avons pris soin de colorer notre ovale avec un dégradé (plus foncé vers le fond pour simuler une sorte d'ombre). On remarquera que notre ligne passe bien à travers notre ovale.

5 Documentation

5.1 Utilisateur

Dans le dossier du projet, vous pouvez y trouver un fichier $KetchupCircle.jar$. Si vous avez java installé sur votre machine, il vous suffit simplement de cliquer dessus. Si cela ne fonctionne pas, ouvrez un terminal et lancer

```
java -jar KetchuppCircle.jar
```

NB : Il est nécessaire d'avoir une version de java ultérieure à java 17. Si l'exécution dans le terminal du fichier .jar vous indique $UnsupportedClassVersionError$, c'est qu'il faut une version plus récente de java. Pour vérifier votre version de java :

```
java -version
```

Si votre machine ne dispose pas de java, procédez à l'installation ici : `https://www.java.com/fr/download/`

5.2 Développeur

Pour ce projet, on a suivi le modèle MVC. Si vous souhaitez reprendre le code, il faut savoir que : la classe principale du Modèle est $Position$; celle de la Vue est $Affichage$, et enfin jeu pour le contrôleur. Nous disposons

également d'un dossier Image dans lequel vous pourrez ajouter d'éventuelles autres images. Toutes les initialisations des threads se font dans le fichier *Jeu*.

Dans le Modèle :
La création et gestion de la ligne brisée se fait dans *Parcours*. Les threads de montée et descente se trouvent respectivement dans les classes *Jump* et *Descendre*.

Dans la Vue :
Une classe *Oiseaux* et *Ciel* pour s'occuper chacun des threads des oiseaux ou des nuages ; et la classe *Affichage* qui s'occupe de tous les affichages à faire (Parcours, Ovale, Oiseaux, Score, Ciel, etc)

Dans le Contrôleur :
Les classes *controleurSouris* et *controleurClavier* qui servent respectivement aux interactions du joueur avec la souris et le clavier ; une classe *controleurBouton* qui gèrent les différentes actions à réaliser selon les boutons. Enfin, la classe principale *Jeu*.

6 Conclusion

Ce projet visant à reproduire le jeu Circle de Ketchapp a été très intéressant. Il m'a permis de mettre en œuvre mes connaissances et mes expériences de travail de groupe dans un projet à réaliser seul.

J'ai pu apprendre à gérer mon temps et mes priorités pour ne pas être trop débordé et en retard. Je suis assez content du ce à quoi mon rendu ressemble, je pense avoir eu de bonnes idées pour rendre mon jeu intéressant à jouer et pas trop punitif, comme par exemple en mettant une barre de vie et une zone rouge lorsque l'on perds de la vie.

J'avais d'autres idées d'implémentations à faire, malheureusement je pense que le rapport temps / bénéfice sur le jeu n'était pas très rentable en vue du peu de temps qu'il me restait pour le faire. J'aurais bien aimé rajouter une option dans le menu qui permettait de changer la couleur du cercle, et pourquoi pas faire une interaction entre le cercle et les oiseaux (par exemple gagner des points si un oiseau passe à l'intérieur du cercle).

Grâce à ce projet, j'ai pu bien mettre en œuvre les threads et en gérer plusieurs en même temps. De plus, c'est la première fois que l'on avait à gérer un aussi grand cahier des charges à faire, et je pense qu'apprendre à faire ça sera un atout majeur pour plus tard.