**Rapport de projet PCII**

1. **Introduction :**

L’objectif de ce projet est de réaliser une version simplifiée d’un jeu très populaire sur smartphone et tablette « flappy bird ». Ici l’oiseau sera remplacé par un ovale se déplaçant le long d’une ligne brisée. Le but du jeu étant que le l’ovale ne sorte pas de la ligne. Le joueur peut cliquer afin de faire monter l’ovale qui redescend ensuite instantanément sous l’effet de la gravité. Graphiquement cela peut ressembler à cela :



2 . **Analyse globale :**

Notre projet se découpe en trois grandes parties :

* L’interface graphique qui représente la vue du joueur où se trouve l’ovale et la ligne
* Le défilement automatique de la ligne brisée donnant l’impression que l’ovale se déplace
* Ainsi que l’interaction entre le joueur et l’ovale (l’ovale monte aux clics de souris)

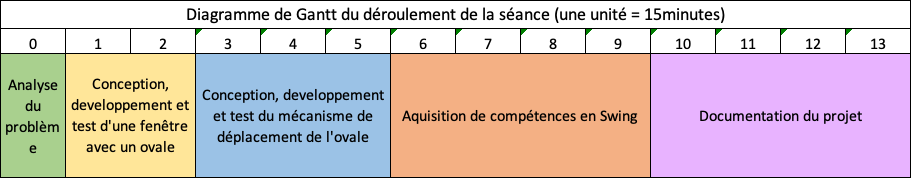
La première partie de ce projet se concentre sur la création d’une fenêtre graphique dans laquelle nous représenterons l’ovale ainsi que l’unique interaction du joueur, c’est-à-dire la montée de l’ovale lorsque l’utilisateur effectue un clic de souris. Ces deux sous partie représente les fondations du projet et sont donc prioritaires. Bien qu’elles soient relativement simples à réaliser.

La seconde partie du projet a pour objectif de générer une ligne brisée aléatoire qui défile et représente le parcours du jeu. Mais aussi la retombée de l’anneaux de façon constante entre chaque clique de souris. Ces deux fonctionnalités sont essentielles pour le jeu et sont donc prioritaires. Elles sont de difficulté moyenne en raison de la gestion des threads afin que les actions s’effectuent en parallèle. Enfin l’algorithme de génération de la courbe est assez difficile à imaginer.

3 . **Plan de développement :**

Première partie :

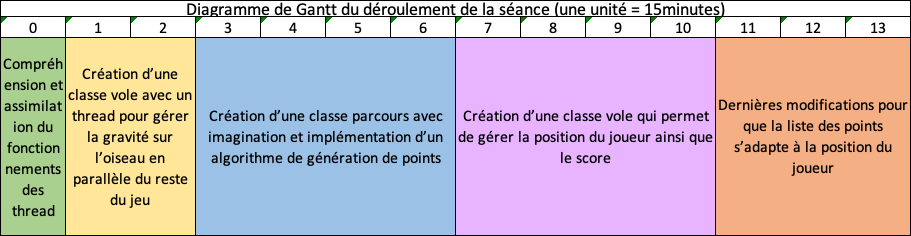
Liste des taches :

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

Deuxième partie :

Listes des taches :

* Compréhension et assimilation du fonctionnements des thread (15min)
* Création d’une classe vole avec un thread pour gérer la gravité sur l’oiseau en parallèle du reste du jeu (30min)
* Création d’une classe parcours avec imagination et implémentation d’un algorithme de génération de points (60min)
* Création d’une classe vole qui permet de gérer la position du joueur ainsi que le score (60min)
* Dernières modifications pour que la liste des points s’adapte à la position du joueur (45min)



4 . **Conception générale :**

Première partie :

Pour le développement de notre interface graphique nous avons reproduit le motif MVC (modèle vue contrôleur).

Notre première sous fonctionnalité au cœur de cette première partie, la fenêtre graphique fait partie de la vue car elle représente une partie de l’affichage. En effet elle définit le cadre dans lequel se déroule le jeu à la fois pour borner l’affichage mais aussi pour d’délimiter le champ d’action du joueur.

La seconde sous fonctionnalité s’inscrit dans la section control car elle fait le lien entre l’interface graphique et le reste du projet. Le mouvement de l’ovale est à cheval entre l’affichage et la modification de l’état du modèle. En effet lorsque l’utilisateur clic alors l’état de l’ovale (centre de l’ovale décalé vers le haut de l’image) est modifié et l’affichage est réactualisé.

Deuxième partie :

Ici l’objectif majeur c’est la création, l’affichage et le défilement de la ligne brisée. Et cela s’inscrit toujours dans un motif MVC.

Nous avons ici implémenté 3 sous fonctionnalités :

La première sous fonctionnalité permet la retombée de l’anneaux entre les clics de souris. Pour cela une classe voler qui hérite de thread a été créer. La méthode « run » du thread permet à notre action de s’effectuer en parallèle du reste de jeu. Une gravité modifie à intervalle régulier la position y de l’ovale pour que celui-ci soit déplacé d’un certain nombre de pixels vers le bas.

La seconde sous fonctionnalité permet la génération d’un ensemble de points afin de créer une ligne brisée. Cela s’effectue dans la classe parcours qui a pour but de généré une liste de points pour la ligne brisée en vérifiant que l’inclinaison de la pente de la ligne ne soit pas trop forte par rapport à la vitesse de retombée de l’anneaux.

Enfin la troisième et dernière sous fonctionnalité permet de gérer l’avancement de la position et du score. Pour cela une classe avancer a été créer avec là encore un héritage de la classe thread pour que l’avancement de la position se fasse en parallèle du reste du jeu. La position est incrémentée de façon constante d’un certain nombre de pixels défini par une constante.

5 . **Conception Détaillé :**

* Concernant la fenêtre avec l’ovale nous utilisons l’API Swing et la classe Jpanel. Les dimensions de l’ovale ainsi que celles de la fenêtre sont définies par des constantes. Pour la fenêtre on a width et height. Et pour l’ovale on a x et y qui correspondent à la position de son centre.
* Pour le déplacement de l’ovale nous utilisons le principe de la programmation évènementielle avec la classe MouseListener et la hauteur définie comme constante. Ici c’est la méthode mouseClicked qui nous intéresse pour permettre la monté de l’ovale à chaque clic de la souris. La constante JUMP définie ici l’amplitude du saut qu’effectue l’ovale. En effet cette constante est ajoutée à la position, en y de l’ovale, à chaque clic de souris.
* La retombée de l’ovale entre les clics de souris est gérée continuellement à l’aide d’un thread dans la classe voler. La méthode « run » applique un délai en milliseconde entre chaque chute de hauteur de l’ovale de la gravité. Un appel à la méthode move de la classe état pour décrémenter la valeur en y du centre de l’ovale d’un certain nombre de pixels vers le bas défini par la constante gravitée.
* La génération de la ligne brisée est gérée par la classe parcours. Un algorithme de génération de points aléatoire avec abscisse croissante est implémenté. Ce dernier prend en charge les bornes de la fenêtre. Il vérifie également que la pente n’est pas trop impotente par rapport à la vitesse de saut (constante jump) et de retombée de l’ovale (constante gravity). Enfin cela prend en compte la position du joueur et permet dans la méthode « incrementPos » d’incrémenté la position et de généré un nouveau point pour la suite de la ligne brisée.
* Enfin la classe avancer utilise un thread afin de gérer l’incrémentation de la position continuellement en parallèle. Elle effectue après un délai un appel à la méthode « incrementPos » de la classe parcours pour modifier sa position. La méthode paint est mise à jour pour permettre l’affichage du score qui est en fait l’état de la position et de dessiner des lignes entre deux points consécutifs du tableau des points de la classe parcours grâce à la méthode drawPolyline.

6 . **Résultats :**

Nous devrions obtenir à thermes l’image présenté en introduction.

7 . **Documentation utilisateur :**

Prérequis : Java avec l’IDE eclipse.

Mode d’emploi : créer un nouveau projet java et déplacer le fichier source à la base du projet. Sélectionner le projet dans la liste à gauche et cliquer sur le bouton « run main » en haut à droite. Une fenêtre s’ouvre et vous pouvez commencez à jouer. Pour fermer le jeu il suffit de fermer la fenêtre.

8 . **Documentation développeur :**

Les prochaines fonctionnalités seront la gestion des collisions lorsque l’anneau entre en contact avec la ligne brisée. Il faudra qu’en cas de contact le jeu s’arrête et affiche le score final. Pour cela nous allons vérifier si l’ovale sorti de la ligne brisée à chaque changement de hauteur (méthodes move et jump).

9 . **Conclusion et perspectives :**

Pour l’instant les bases du projet ont été posée. La structure du projet a été établie et les différents outils nécessaires ont été identifié et apprivoisé. La fenêtre graphique a été réalisée et avec elle les conditions d’affichage et d’exécution de projet. L’ovale représentant le joueur a été créer ainsi que son action de base : le saut. La ligne brisée a été généré et ainsi que son défilement. L’ovale est maintenant soumis à une gravité. De plus la position du joueur à été implémenté et s’incrémente continuellement en parallèle du jeu. Enfin le score est calqué sur la position et s’affiche dans la fenêtre.

La prochaine séance sera consacrée à la gestion des collisions entre l’ovale et la ligne brisée.