Premier Cycle
Mini-Projet Informatique 2ème année
2018 - Semestre 4

INSA INSTITUT NATIONAL DES SCIENCES APPLIQUÉES LYON

Rendu
-
« Wings Up »

I. Nom des participants:

Salima Makhloufi	71	Jean Haberer	71
Inès Lebouteiller	71	Amaury Chapelle	71

II. Nom du Projet : Wings Up

III. Cahier des charges

Le but de ce projet est de développer un jeu reprenant le principe de « Dune » ou encore « Tiny Wings ».

Les oiseaux partent de leur nid, en haut d'une dune et doivent s'élancer en s'aidant de la courbure des dunes et de leur poids en appuyant sur une touche. Plus la trajectoire est proche de la courbure de la dune plus le saut sera réussi et l'oiseau ira loin.

Il faut alors appuyer sur une touche pour descendre et retomber sur la dune et relâcher quand on veut s'envoler.

IV. Méthode choisie : explication et justification

Nous avons choisi de diviser le programme en 4 objets. L'objet Jeu est créé dans le main et il crée à son tour les objets Bird, Dune et Renderer.

Les deux objets qui interagissent le plus ensemble sont la dune et l'oiseau car l'oiseau récupère la position de la dune pour savoir quand est-ce qu'il touche le sol ou quand est-ce qu'il est dans les airs. Le Renderer est un JPanel qui sert à dessiner le jeu de manière fluide.

Ensuite nous avons réfléchi aux différents attributs et méthodes que ces objets devaient avoir pour satisfaire leur rôle. Nous avons alors divisé en deux le travail, les méthodes liées à l'affichage du jeu et les méthodes liées à l'interaction et au calcul des positions de la dune.

V. Programmation : explication et justification

Afin d'expliquer le programme, nous allons traiter chaque objet séparément :

V.1. Objet : Dune

V.1.1. Méthode : Refresh

Type : **void**

Attribut : **int Vitesse**

Décalage des pixels un par un dans un tableau d'entiers. On décale plus ou moins selon la vitesse d'entrée.

Calcul des prochains points, on utilise la fonction cosinus pour obtenir des vagues tout du long. On donne une pente aléatoire et une largeur aléatoire pour avoir cet effet naturel. Et on enregistre ce point dans le tableau. On inverse ensuite le sens de la pente pour avoir des pentes ascendantes et descendantes alternativement.

V.1.2. Méthode : Draw

Type : **void**

Attribut: **Graphics g, bird Bird, int ax, int lx, Boolean gameOver**

Tant que l'utilisateur n'a pas perdu, on appelle **Refresh** avec une valeur dépendant de la vitesse de l'oiseau, donnée par **cx** - **lx**.

En parallèle, on dessine le ciel avec un gradient de couleur changeant qui occupe toute la fenêtre.

Marquage du point central définit par la position verticale de l'oiseau, ce point doit être au milieu de la fenêtre.

Ensuite on dessine la dune rectangle par rectangle ayant pour abscisse l'indice du tableau **courbe**, pour ordonnée celle du centre et la taille de la dune, comme largeur 1 et comme hauteur la différence entre la hauteur totale de la fenêtre et l'ordonnée.

Actualisation de la fluctuation des couleurs de la dune et du ciel.

Si l'utilisateur a perdu, on appelle **Refresh** avec la valeur 1 pour que la dune se dessine encore à une vitesse lente.

V.1.3. Méthode : Setup

Type : **Void**

Sans attribut

Initialisation de l'affichage de la dune pour le début du jeu ou il y a un plat et une première descente. C'est l'équivalent de la méthode **refresh** mais appelée seulement au début.

V.2. Objet : Bird

V.2.1. Méthode : Refresh

Type : **int**

Attribut : **Dune dune**

Récupération de la position du sol contenu dans le tableau de l'objet **dune**. Calcul de la pente de la **dune** à partir du tableau **courbe**.

Traitement de tous les cas possibles d'interaction entre l'oiseau et la dune en fonction de la pente et la position du sol par rapport à l'oiseau.

Le graphe ci-dessous nous facilite la compréhension de cette méthode :

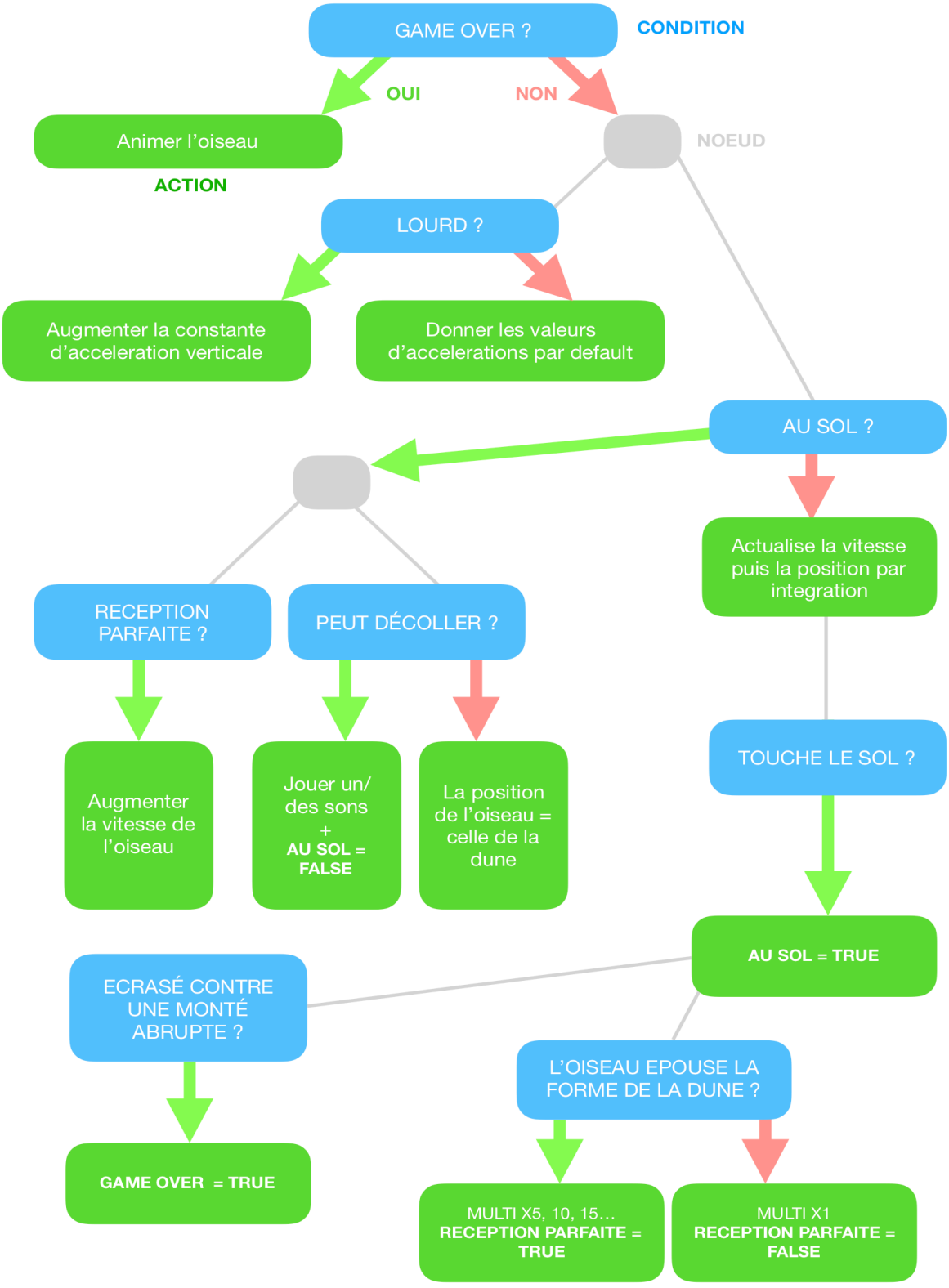

Figure 1 - Diagramme résumant les conditions if

V.2.2. Méthode Draw

Type : **boolean**

Attributs: **Graphics g, boolean writing**

Récupération des informations concernant la rotation de l'oiseau. Dessin de l'oiseau avec l' inclinaison qui correspond. Ecriture du score avec le coefficient multiplicateur si la réception était parfaite.

Renvoi du booléen **true** ou **false** suivant la valeur de **gameOver** de l'oiseau.

V.2.3. Méthode Touched

Type : **void**

Attribut : **boolean b**

Cette méthode change la valeur de **heavy** en **true** ou **false** suivant si l'oiseau a été touché ou non. Il permet d'interagir avec l'utilisateur via l'objet **Game** qui possède les action listener.

V.3. Objet : Game

V.3.1. Constructeur

V.3.2. Méthode : Repaint

Type : **void**

Attributs : **Graphics g**

Elle appelle les méthodes **draw** de **dune** et de **bird** définies précédemment.

Selon si le joueur a perdu (**gameOver**) ou s'il joue les différents textes sont affichés à l'écran. Si **gameOver** est vrai alors elle affiche le score fait avec la possibilité de rejouer, de voir les instructions ou de quitter.

Si le joueur est en train de jouer, elle passe la musique du jeu et la réinitialise. Elle affiche aussi le score de l'utilisateur.

Si menu est vrai alors elle affiche le menu avec les mêmes commandes permettant de commencer une partie, quitter ou voir les instructions.

Si le joueur appuie sur pause, la méthode affiche pause à l'écran.

V.3.3. Méthode : Override keyPressed

Type : **void**

Attribut : **KeyEvent e**

On associe à chaque touche du clavier une action :

Pour le « p », cela active le booléen pause ou le booléen **playingGame**.

Pour la lettre « m » cela change la valeur du booléen **menu** et gère les différents sons.

Pour la lettre « i », cela affiche les instructions à l'aide d'un **JOptionPane Message dialog**. Les instructions à suivre ont étés saisies à l'intérieur de cette boite de dialogue.

Pour le « q », cela quitte le système.

Pour toutes les autres touches du clavier, c'est le poids de l'oiseau qui est affecté. Donc la méthode **touched** de **Bird** est appelée.

V.4. Objet : Renderer

V.4.1. Méthode : paintComponent

Type : **Void**

Attributs : **Graphics g**

Elle appelle la méthode **Repaint** de **Game** pour faire le rendu.

VI. Hiérarchie : Diagramme UML

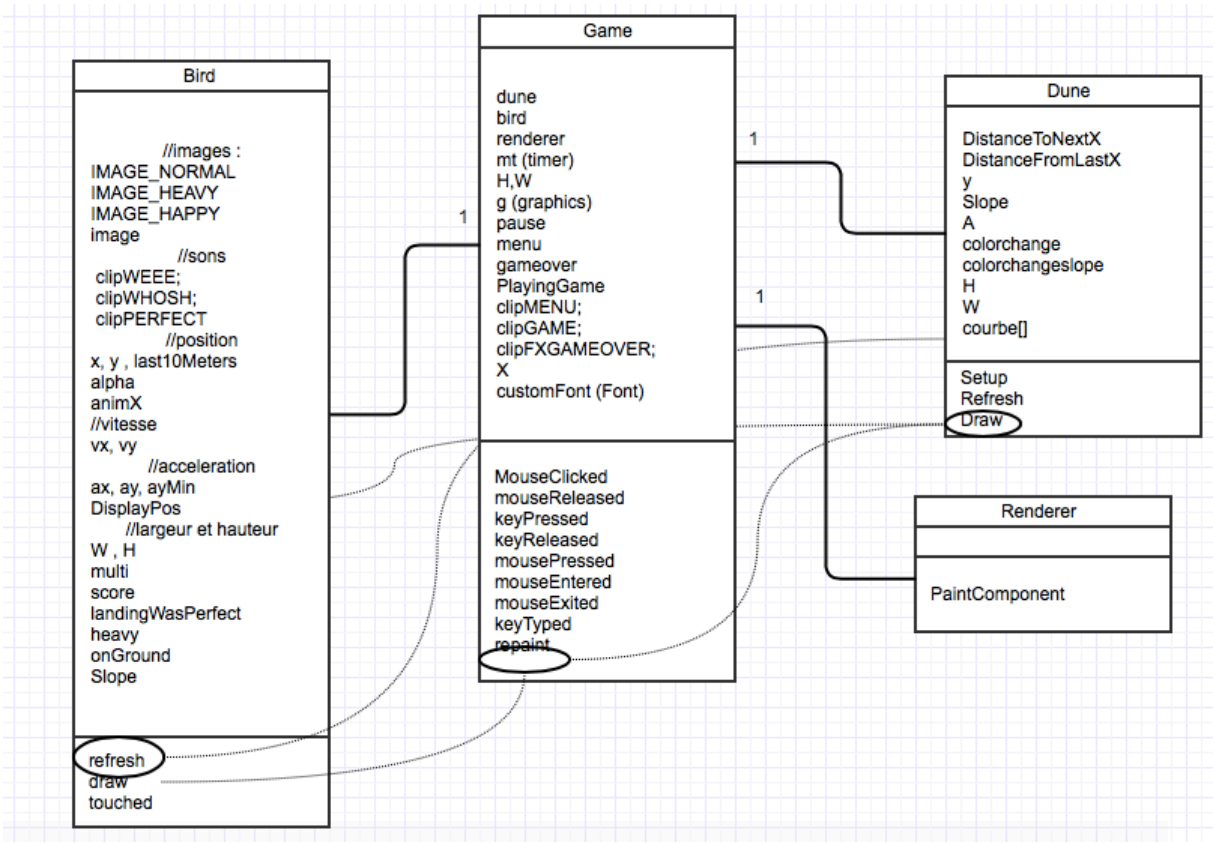

Figure 2- Diagramme UML

VII. Problème physique

L'oiseau suit des paraboles en s'élançant du haut d'une dune. Il prend de la vitesse dans les pentes, et lorsque l'utilisateur relâche la touche permettant de jouer alors que l'oiseau arrive au sommet d'une dune, celui-ci décolle et vole sur une certaine distance en fonction de la qualité du saut réalisé.

VIII. Principe de l'algorithme

La partie mécanique du mouvement est traitée dans la classe Bird, il s'agit de la méthode « refresh », prenant comme paramètre un objet Dune.

La méthode est constituée d'une imbrication de if/else. Elle traite différentes situations, pour commencer, elle traite le cas « le joueur n'a pas perdu ». Dans ce cas-là, on définit deux composantes pour l'accélération, celles-ci peuvent être considérées comme étant des caractéristiques de l'oiseau, que l'on fera varier en fonction de la position de l'oiseau. Si l'utilisateur appuie sur la barre espace, il cherche à ce que l'oiseau prenne de la vitesse dans une descente. Pour cela, on cherche à augmenter son accélération, on modifie donc les valeurs des deux composantes en x et en y de l'accélération.

IX. Améliorations possibles

Quelques améliorations peuvent être apportées à notre projet, ceci dit on pourrait faire deux modes de jeu : "Vol solo" et "Nuée d'oiseaux", autrement dit : soit jouer tout seul ou **jouer en groupe**. D'autre part, afin d'améliorer les performances de l'algorithme, on aurait pu notamment utiliser un tableau dynamique pour éviter le **recopiage** du tableau courbe. En effet notre algorithme utilise un O(N) pour effectuer tout le décalage des copies alors qu'avec un tableau dynamique cette opération pourrait être rendu beaucoup rapide. Une « **linked list »** permettrait d'ajouter une valeur et redéfinir la première en conservant celles du milieu sans décaler toutes les cases.

X. Echéancier :

Taches	Fevrier	Mars				Avril
	Mardi le 27	Mardi le 6	mardi le 13	mardi le 20	mardi le 27	du 1 er au 5
Choix du sujet et Validation par l'enseignant						
Préparation de cahier de charge						
Determination de la hierechie des classes						
Determination des attributs						
Ecriture des methodes						
Recherche des graphismes(image , police)						
Description physique du probleme						
Ecriture du rapport final						

XI. Bibliographie

Les images IMAGE_NORMAL IMAGE_HEAVY IMAGE_CONTENT sont issues du jeu **ANGRY BIRDS** qui appartiennent à la société **ROVIO ENTERTAINMENT**. La police utilisée dans le jeu est une police gratuite appelée **Minercraftory**. Les sons **WEE**, **WHOSH**, **PERFECT**, **MENU**, **GAME** et **GAMEOVER** sont libres d'utilisation non commerciale et trouvés sur youtube.