Loïs Aubree - Lucie Boutou
Benjamin Guillet - Théophile Madet

Projet de LO43

Réalisation d'un jeu AngryBirds-like

Rapport de projet

Automne 2011

Table des matières

Chapitre 1

Présentation du projet

Nous avions ce semestre en LO43 la possibilité de développer un jeu ludique similaire à Angry Birds.

Dans notre version, le joueur a à sa disposition un ensemble d'oiseaux «en colère» , et doit tuer les ennemis à l'aide d'oeufs pondus en vol. Les oiseaux peuvent également se sacrifier comme dans Angry Birds en réalisant une «attaque suicide» et tuer les ennemis avec leur propre corps.

Le joueur a au départ entre 3 à 5 oiseaux, selon la difficulté du niveau. Chaque type d'oiseaux possède des capacités de ponte (nombres d'oeufs) et des possibilités physiques qui lui sont particulières. Le score du joueur est calculé d'après son nombre d'oiseaux restants à la fin du niveau. Celui-ci est fini quand tous les ennemis sont morts.

Le jeu est organisé en suivant des difficultés qui vont de facile à extrême, et chacune d'elle possède un nombre défini de niveaux. Seul le niveau 1 de chaque difficulté est au départ disponible, les niveaux suivants sont débloqués au fur et à mesure de la progression du joueur. Ce dernier ne peut accéder au niveau suivant qu'en ayant validé le précédent.

Chapitre 2

Organisation et répartition du travail

Notre groupe était composé de quatre personnes, nous avons régulièrement organisé des réunions chez les uns et les autres pour avancer le projet et faire le point.

La partie spécification UML a été faîte une première fois tous ensemble afin de se mettre d'accord sur une base de départ puis a évolué pour mieux coller au cahier des charges : respect du pattern MVC, utilisation des threads et du polymorphisme etc.

Nous avons utilisé Eclipse, Skype et un dépot GitHub pour la synchronisation et la gestion des versions. GitHub possède également un suivi des fonctionnalités à ajouter ou à corriger. Ce qui a permit à chacun de travailler les classes et fonctionnalités nécessaires qui lui plaisaient.

Globalement on peut dire que le travail s'est réparti ainsi :
- Benjamin : base du projet (respect du pattern MVC), gestion du joueur ;
- Loïs : collisions et envoi des oiseaux ;
- Lucie : décors et menus ;
- Théophile : gestion des sauvegardes, collisions.

Mais tout le monde a par moment débordé sur le «secteur» des autres pour corriger un bug ou améliorer une fonctionnalité. Rien n'était figé.

Chapitre 3

Spécification

3.1 Diagramme de classes

3.2 Diagramme de séquences

Chapitre 4

Conception et prise en main

4.1 Gestion du menu

La gestion du menu s'effectue au travers de 6 panneaux différents :
- MenuHomeView, panneau d'accueil au lancement du jeu.
- MenuNewView, panneau de création d'une nouvelle partie.
- MenuLoadView, panneau de chargement d'une partie.
- MenuControlsView, panneau d'informations sur les contrôles du jeu.
- MenuDifficultyView, panneau de choix de la difficulté du jeu.
- MenuLevelView, panneau de choix du niveau.

Chaque panneau est implémenté dans une classe propre à ses caractéristiques héritant de la classe GameViewMenu qui comporte les caractéristiques communes à tous les panneaux telle que le background du Menu.

La classe GameViewMenu hérite elle-même de la classe JLayeredPane, qui permet d'indiquer aux éléments placés sur le panneau un index de position en profondeur, ce qui nous permet notamment de placer nos boutons par-dessus notre background.

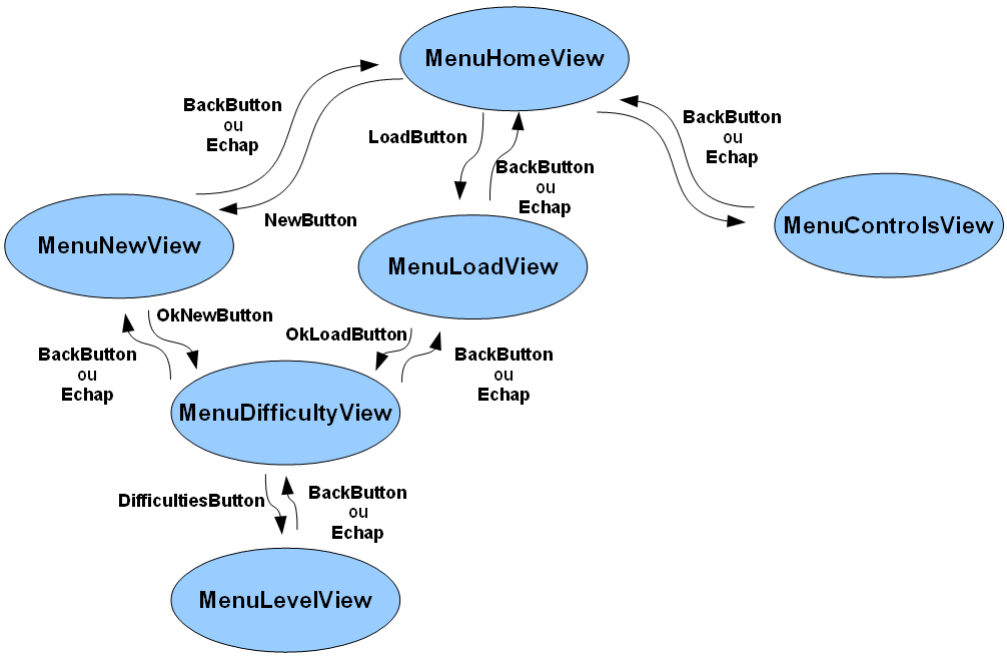

FIGURE 4.1 – Navigation dans le menu

La navigation dans le menu se fait au travers des différents boutons avec la souris, et pour le retour en arrière nous avons mis en place un contrôle supplémentaire par le clavier au moyen de la touche «échap »(équivalent au retour en arrière avec le bouton «retour »à la souris).

Il y a alors détection du panneau sur lequel on se trouve afin de charger le panneau «précédent »correspondant.

Pour le cas du panneau de choix de difficulté, l'accès au panneau a pu être réalisé depuis le panneau de création d'une nouvelle partie ou depuis le chargement d'une partie. C'est pourquoi une variable a été implémentée (parentPanel) afin de garder en mémoire depuis quel panneau a été fait le dernier accès au panneau de choix de difficulté.

4.2 Gestion des niveaux

La création d'un niveau du jeu s'appuie sur la lecture d'un fichier texte qui doit contenir les informations suivantes :
- les oiseaux disponibles pour réaliser le niveau.
- la position des différents blocs représentant le décor.
- la position des cochons dans ce décor.

On retrouve en tête du fichier la liste des oiseaux. Un retour à la ligne est effectué après chaque oiseau disponible dans le niveau afin de faciliter la lecture du fichier. De plus les oiseaux sont indiqués dans le même ordre que leur ordre d'apparition dans le niveau.

Un test d'égalité a lieu entre la chaîne de caractère contenue dans la ligne et les différents types d'oiseaux, afin de créer l'oiseau correspondant et de l'ajouter à un ArrayList d'entités.

Lors de la lecture du fichier texte, la rencontre du mot «Map »permet au programme de savoir que la liste des oiseaux disponibles pour le niveau est complète et que la suite du fichier contient la carte représentant le niveau.

Il s'agit de lignes de texte de taille égale contenant différents caractères qui seront implémentés dans un tableau de deux dimensions (chaque ligne du fichier texte correspond à une ligne du tableau 2D).

Le tableau à une taille adaptable aux dimensions de notre fenêtre et de nos éléments de décor. Chaque élément du décor a une image correspondante de 26*26 pixels, et pour correspondre ici à une fenêtre de taille 1200*600 pixels, une ligne du tableau contient 47 éléments et une colonne en contient 22.

Chaque caractère correspond à un élément différent sur le décor :
- le «0 »indique que rien ne se trouve à cet endroit de la carte.
- le «1 »indique qu'il y a un bloc d'herbe à cet endroit de la carte.
- le «2 »indique qu'il y a un bloc de pierre à cet endroit de la carte.
- le «3 »indique la position de départ d'un cochon ennemi.

Une lecture du tableau est ensuite réalisée et à chaque «1 », «2 », ou «3 »rencontré, on créé l'entité correspondante (cochon, bloc d'herbe ou bloc de pierre). L'entité est créé via un constructeur prenant en paramètre sa position x (calculée à l'aide de l'index des colonnes du tableau et de la taille de l'image correspondante à l'entité) et sa position y (calculée à l'aide de l'index des lignes du tableau et de la taille de l'image correspondante à l'entité).

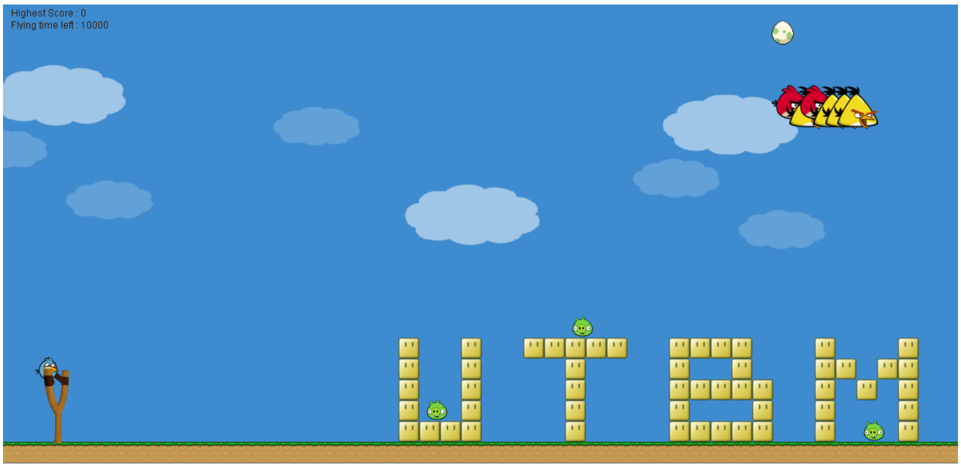

FIGURE 4.2 – Screenshot niveau 1

```
Sparrow
Humming Bird
Pigeon
Humming Bird
Pigeon
Pigeon
Pigeon
Map
00000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000
00000000000000000000000000030000000000000000000
00000000000000000002002002222200222200020002000
00000000000000000002002000020000200200022022000
00000000000000000002002000020000222220020202000
00000000000000000002302000020000200020020002000
00000000000000000002222000020000222220020302000
11111111111111111111111111111111111111111111111
```

FIGURE 4.3 – Fichier texte niveau 1

La figure 4.2 est le fichier texte correspondant au niveau représenté en figure 4.1. On remarque que la dernière ligne ne comportant que des «1 »concorde bien à

une ligne de blocs d'herbe sur la partie basse de notre fenêtre. De même les «2 »concordent à des blocs de pierres et les «3 »à des cochons.

On aperçoit dans le coin en haut à droite de la figure 4.1 les oiseaux disponibles dans le niveau, coïncidant aux premières lignes du fichier texte de la figure 4.2.

D'une manière générale on remarque que la création de nouveaux niveaux ou la modification de niveaux existants est relativement simple.

Chapitre 5

Bilan

5.1 Conclusion

Ce projet était très intéressant et motivant. Son côté ludique nous a permis d'apprendre le java avec une vraie motivation et une bonne ambiance. Un groupe important comme le notre nécessite une bonne communication afin que chacun sache le travail à accomplir et éviter les répétitions. Heureusement aujourd'hui nous avons de nombreux outils à notre disposition pour nous faciliter la tâche, en plus des classiques réunions ou e-mails.

5.2 Améliorations possibles

Un projet comme celui-ci est toujours perfectible, voici quelques-unes des améliorations qui nous viennent en tête et que nous aurions bien voulu implémenter :

- Il serait facile d'ajouter une musique d'ambiance, des bruits lors des collisions ou lorsque l'on gagne ou perds ;
- On pourrait afficher le meilleur score du joueur actuel sous chaque niveaux ;
- Les ennemis pourraient gérer la gravité, c'est à dire s'ils sont placés sur un bloc en hauteur, descendre jusqu'en bas lors de la destruction de ce bloc.