Romaric MOLLARD - Guillaume RUCHOT

5 pages

POUR VOIR L'INTÉGRALITÉ DU JEU, METTEZ LA VARIABLE *DEBUG* À *TRUE*

Introduction

La programmation orientée objet change considérablement notre manière de concevoir nos applications. Tous les domaines ne nécessitent pas une utilisation de ce concept, le jeu en revanche est l'un des domaines les plus aptes à son utilisation. En effet comme son nom l'indique ce type programmation met en valeur la décomposition en objets de l'application, et un jeu contient le plus souvent une multitude de composantes. On peut ainsi aisément découper le code en chacun des objets qui vont le composer. Réaliser un jeu est donc le meilleur moyen de comprendre les principes de la programmation objet.

AntsVsSomeBees est un jeu composé de trois éléments principaux le *terrain*, la *fourmi* et l'*abeille*. Lors de la réalisation du jeu nous avons pu mettre à jours des objets supplémentaires comme l'objet *son, image, animation, fenêtre*, du côté moteur de jeu ; et des objets comme la *place* et la *colonie* du côté de la gestion du terrain.

Présentation du jeu

Notre *AntsVsSomeBees* est une large amélioration du projet initial. Nous avons fait plusieurs choix quant au fonctionnement du jeu.

Premièrement c'est un jeu de type arcade, les vagues d'ennemi arrivent en continue et le but du jeu est de tenir le plus longtemps et d'amasser le maximum de points (XP) pour battre le record (Best).

Ensuite nous avons décidé de rendre le jeu progressif en débloquant peu à peu les tunnels et les fourmis utilisables.

Le jeu commence donc avec les fourmis de base :

Queen - Au début du jeu elle n'est pas ou peu importante, la reine permet d'augmenter les fonctionnalités des fourmis devant et derrière elle, ainsi une *Harvester* produira deux fois plus de nourriture par tour en étant au contact de la reine. Cependant une fois placée, la reine ne peut plus être déplacée, et sa mort entraine une grande explosion qui détruit la colonie.

Harvester - Elle crée simplement la nourriture indispensable à l'achat de fourmis.

Thrower - C'est le soldat de base du jeu, elle tire à une distance maximum de 3 cases et une fois par seconde. Elle fait 3 dégâts quand elle n'est pas à côté de la reine.

Les fourmis de base permettent de repousser les premières vagues pour ainsi débloquer les prochaines fourmis. Nous n'allons pas détailler chacune des fourmis et augmentations mais en voici quelques unes :

Wall - Cette fourmi ne tire pas, mais possède beaucoup de vie, elle peut donc tenir les abeilles à l'écart jusqu'à sa mort.

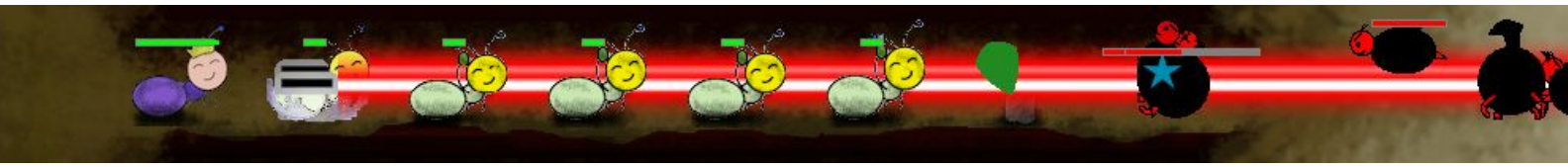
Stun - Cette fourmi avec la *Slow* permet de ralentir les abeilles, mais ne fait aucun dégât.

Guardian - Cette fourmi (avec son amélioration) se place sur une autre fourmi, la protégeant des monstrueuses abeilles.

Buff - Cet organisme, composé de microscopiques reines, se place sur une fourmi pour la galvaniser, de la même manière que la *Queen*. (Les *Guardians* étant fidèles à leur reine, ils refuseront de protéger quiconque utilise ce substitut).

Nénuphar - Cette invention, créée par l'ingénieur *NewtAnt* , permet à des fourmis terrestre de tenir sur l'eau. Cette idée lui serait venu de la part d'un ami , *JésAnt* .

TAntK - *EinstAnt*, le rival de *NewtAnt,* créa cette arme de guerre onéreuse mais tirant des lasers capables d'infliger de lourds dégâts à toutes les abeilles sur une ligne droite. En temps de paix, elle font de très beaux feux d'artifices.



Modifications apportées

Nous avons rajouté une multitude d'ajouts graphiques en plus des modifications du fonctionnement. Voici une liste des ajout depuis la version de base.

- Système de statistiques et de points
- Affichage de l'arrivée des abeilles à l'avance
- Sauvegarde du record
- Affichage de la création de nourriture
- Affichage des explosions
- Gestion du son et des musiques
- Défilement du menu de fourmis (appuyer et bouger la souris pour défiler le sélecteur de fourmi)
- Système d'arcade
- Menu et générique de début
- Ajout de 4 niveaux d'abeille

- Ajout du nénuphar pour recouvrir les cases d'eau
- Ajout de l'amélioration unique, à poser sur une fourmi pour augmenter ses capacité
- Animation des abeilles et des fourmis (respiration, oscillation, battement d'ailes...)
- Amélioration des graphismes pour plus d'homogénéité
- Ajout de plusieurs vies à la colonie
- Ajout du texte au survol des objets
- Ajout de lasers pour la TantK
- Ajout des barres de vie pour les fourmis et leurs armures
- Ajout de tombes lors de la mort des fourmis
- Ajout d'images pour les fourmis galvanisées, et pour la *Hungry* après son repas
- Ajout de références.

Liste des fourmis présentes : *Queen, ShortThrower, Thrower, LongThrower, Guard, BetterGuard, Stun, Slow, Fire, BetterFire, Harvester, Ninja, Hungry, TantK, Scuba, Wall, Buff, Waterlily.*

Difficultés rencontrées

Il a tout d'abord été difficile de s'adapter au squelette fourni, car la classe *antGame* par exemple était très longue et l'on devait comprendre le fonctionnement des nombreuses fonctions la composant, pour pouvoir par la suite les utiliser et les modifier. Après plusieurs heures de lecture et de tests, nous avons pu saisir le programme dans sa globalité, et enfin commencer à ajouter les premières fourmis demandées.

Très rapidement, nous nous sommes confrontés au second problème, qui sera resté jusqu'au bout du projet le plus conséquent : le squelette qui nous a été fourni n'était pas souple du tout, et ainsi la moindre modification au coeur du jeu entraînait de très nombreuses complications, dont certaines insolvables sans changer énormément la composition du squelette, ce qui allait à l'encontre de l'un des objectifs du projet , c'est à dire l'adaptation à un code déjà existant. Ce problème n'a pas pu être résolu dans sa globalité, mais tout les problèmes qu'il a entraîné ont été résolus soit en faisant de nombreux tests et en corrigeant petit à petit chaque erreur qui se présentait, soit en abandonnant certaines idées que l'on a jugés irréalisables.

Parmi les exemples de ces difficultés, on peut en mentionner deux en particulier :

- L'ajout d'une fonction de restart , pour pouvoir recommencer une partie sans avoir à relancer le programme. Cette fonctionnalité nous posait beaucoup trop de problèmes (bugs graphiques notamment), et après de nombreuses approches différentes, nous avons fini par l'abandonner totalement.

- L'ajout d'un générateur de jeu infini. Dès le début nous n'aimions pas le caractère limité du jeu à un nombre limité d'abeilles, avec une durée de vie très très courte et très peu de rejouabilité. Ainsi nous avons pensé le transformer en un pseudo-jeu d'arcade, et cette transformation fût très difficile mais nous avons persévérés et finis par surmonter ces difficultés. Tout autant que nous voulions ajouter un visuel des abeilles avant leur arrivée, ce qui demandait une modification totale du fonctionnement de la classe *Hive*.

D'autres difficultés que nous avons rencontrés furent la gestion de la reine des fourmi, et celle des vies de la colonie. Chacune de ces deux implémentations prise indépendamment n'étaient pas très dure à faire, mais les deux ensembles posaient de sérieux problèmes : faire attention que la reine ne meure pas avant la colonie, synchroniser leurs deux barres de vie, … . Encore une fois, pour résoudre ces difficultés de nombreux tests et de nombreuses petites modifications à l'ensemble du code se sont révélés nécessaire.

Temps passé sur le projet

	Guillaume	Romaric
Conception	5h	5h
Codage	30h	35h
Tests	5h	6h
Son	-	2h
Graphismes	3h	3h
Documentation	5h	-
Rapport	2h	2h
Total	**50h**	**53h**

Bon jeu !