FOLLET Hugo L2 Info
CAPITAINE Robin
CONSTANT Clément

Projet de Java : Zaq

Notre jeu, Zaq, est un shoot 'em up inspiré de jeux tels que Space Invader voire Touhou. Le joueur contrôle un petit slime blanc nommé Zac, dont l'image est fortement inspirée de Minecraft, qui va détruire les autres monstres du jeu cubique qui ne l'aiment pas. Ainsi, sur un total de 4 niveaux et d'un dernier niveau contenant une unique phase de boss, le joueur pourra essayer d'abattre l'ensemble des ennemis sur tous les niveaux.

Le gameplay du jeu se divise ainsi en 5 niveaux, présents dans le dossier levels et tous des sous-classes de Level.java. Les 4 premiers possèdent respectivement 1 à 4 vagues d'ennemis, et chaque vague contient 1 à 4 ennemis, présents dans le dossier ennemis et tous des sous-classes de enemmi.java. Ces ennemis se divisent en 3 groupes :

-Les Slimes Verts, du fichier SlimeVert.java, sont des ennemis relativement faibles qui se déplacent de gauche à droite en ligne droite et tirent en direction du joueur.

-Les Pillards, du fichier Pillard.java, sont des ennemis un peu plus résistants qui se déplacent de gauche à droite en sinusoïde et tirent 3 balles en même temps.

-Le Withers, du fichier Wither.java, sont des ennemis très résistants qui se déplacent de gauche à droite en sinusoïde et tirent en cercle.

Le cinquième niveau, celui du boss, reprend dans son unique vague quelques Withers, un Slime et inclut le boss, l'EnderDragon, du fichier EnderDragon.java. Ce dernier est extrêmement résistant et tire avec 3 armes en simultané. Le joueur passe ainsi de vague en vague dans chaque niveau, puis de niveau en niveau automatiquement grâce à la fonction checkLevels() du fichier ArianeGame.java.

Nos ennemis possèdent chacun une arme différente trouvable dans le dossier "Weapon", elle donne à leur balle leur puissance et gère la fréquence de tir. Évidemment chaque ennemi, tout comme le personnage que nous incarnons, possède de la vie. Durant une partie, la barre de notre personnage sera affichée dans une barre hors-jeu, visible en haut de l'écran, définie dans le dossier "hud". Ce ne sera pas le seul à voir sa barre de vie affichée puisqu'une nouvelle barre violette sera visible pendant le jeu, celle-ci sera la barre de vie du boss qui s'affichera quand nous nous trouverons contre celui-ci.

De plus les ennemis ont une chance sur deux de faire apparaître un boost, chaque boost étant une sous-classe de LuckyBlock.java. Ces LuckyBlocks descendent progressivement jusqu'en bas de la fenêtre de jeu et disparaissent lors d'une collision avec le joueur ou la bordure inférieure. Pour récupérer le power-up, le joueur doit toucher le LuckyBlock, sinon ce dernier est perdu.

Ainsi, lors de l'apparition d'un boost, il y a deux chances sur trois que ce dernier soit un bonus de vitalité, provenant de Heal.java. Ce dernier peut soigner jusqu'à 200 points de vie au joueur, mais peut également lui en faire perdre jusqu'à 50, c'est ainsi une lame à double tranchant. Le reste des probabilités est une mise à niveau de l'arme du joueur, via le fichier UpWeapon.java. Lors de l'obtention du bonus, l'arme du joueur se verra être remplacée par une arme plus puissante.

Pour motiver le joueur à faire mieux que les autres fois, un système de score a été établi, défini dans le dossier "hud". Il sera également visible dans la barre hors-jeu en haut de l'écran et permettra au joueur de suivre son score en temps réel. Pour augmenter le score, il faut donc tuer des

monstres, le but étant de battre le record visible en bas à gauche de l'écran, défini dans le fichier "Score.java".

Pour jouer au jeu, nous avons décidé de faire une véritable interface permettant d'utiliser des fonctionnalités différentes que nous pouvons retrouver dans d'autres jeux. En lançant le jeu nous tombons donc sur un menu qui nous propose 4 fonctionnalités : Play, pour lancer directement le jeu, Select, pour nous sélectionner notre niveau, High-Score, pour voir notre record, et Quit pour fermer le jeu.

Le fichier ArianeGame.java possède, dans la fonction update(), une possibilité de mise en pause du jeu. Cette pause se définie en 2 parties. La première est l'application de la pause. Si le jeu est en pause, il n'y a aucune mise à jour des entités possible, on vérifie donc que la pause n'est pas enclenchée pour mettre à jour le joueur et les ennemis. La seconde partie est l'activation et désactivation de la pause. Lorsque le joueur appuie sur Entrée, le programme va vérifier si le jeu est en pause. Si oui, il va remettre le curseur du joueur au bon endroit, le faire disparaître et relancer les mises à jour. Si non, il va bloquer les mises à jour, enregistrer la position du joueur et afficher à nouveau le curseur.