Université de Bordeaux

Architecture Logicielle

STR et framework de jeux 2D

Auteurs :
Elyas Ben Hadj Yahia
Ryan Herbert
Paul Maribon-Ferret

Enseignant :
David Auber

31 janvier 2014

Table des matières

1 | Présentation

1.1 Fonctionnalités du jeu

Ce jeu est un mélange entre les jeux de types Tower Defense, et les jeux Zelda. Le joueur a le contrôle de Link et a accès à tous les endroits du terrain de jeu, mis à part quelques obstacles comme les murs ou comme les flaques d'eau. En revanche, les *creeps* sont restreints aux terrains de type "Path", comme dans certains Tower Defense, et ne peuvent en aucun cas quitter le chemin, et il est impossible de les y contraindre.
Il existe une condition de défaite : lorsque Zelda (destination finale des *creeps*) n'a plus de points de vie, la partie est perdue. La condition de victoire est d'éliminer tous les *creeps* que les Spawns prévoient d'envoyer à l'assaut de la princesse. Link n'est lui même pas invulnérable, il possède un certain nombre de points de vie aussi. S'il percute un mur (en se faisant éjecter dans un mur au contact d'un *creep*), il ne perdra pas de points de vie, mais il sera renvoyé à son point d'apparition initiale. Si ses points de vie atteignent zéro, il sera également renvoyé à son point d'origine, et son compteur de vie sera initialisé à trois.
Link possède deux équipements : son épée, et son bouclier.
Lorsque Link équipe son épée, il peut frapper les *creeps*, ainsi les repoussant, et leur infligeant des dégâts.
Lorsque Link équipe son bouclier, il peut repousser les *creeps*, de la même manière qu'avec l'épée, mais ce sans infliger de dégâts.

2 | Architecture

2.1 Modifications du framework

Afin de pouvoir définir des tailles de sprite différentes et rectangulaires, nous avons créé notre propre classe OctolinkSpriteManager. Cela nous a également permi de créer des bounding box avec des tailles personnalisées et de bien les aligner par rapport aux sprites. Nous avons été obligé de dupliquer et modifier la classe IntersectTools du framework existant pour gérer les collisions correctement avec nos bounding box personnalisées.

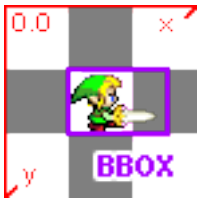

FIGURE 2.1 – Link sprite bounding box

De plus, pour rajouter une barre de vie pour Zelda et implémenter les fonctions *pause* et *resume*, nous avons été contraints de dupliquer les classes GameDefaultImpl et GameLevelDefaultImpl.

2.2 Architecture du jeu

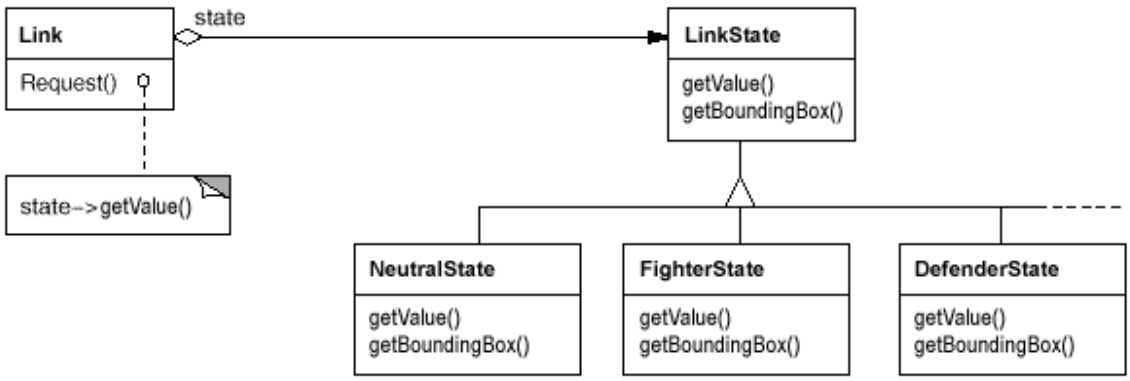

FIGURE 2.2 – State pattern

En utilisant le pattern State, nous avons pu gérer les différents états de Link, notamment en déléguant toutes les opérations spécifiques à l'état.
Ceci nous a facilité la gestion des bounding box différentes et des collisions en fonction de l'état.

2.3 Problèmes rencontrés

- La modification des bounding box engendrée par l'utilisation de sprites de tailles différents a été complexe car il a fallu intégrer les modifications au système de gestion des collisions.

- L'implémentation des fonctions *pause* et *resume* s'est avérée assez délicate à cause de la gestion des Threads avec les méthodes *wait()* et *notify()* en étant *synchronized*.

3 | Annexe

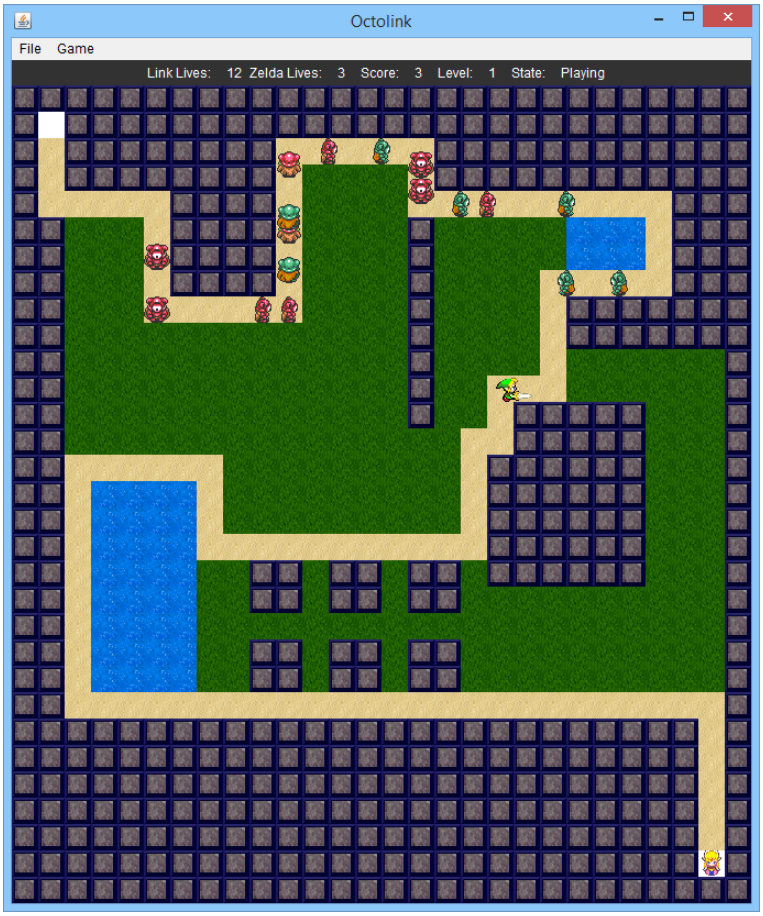

FIGURE 3.1 – Screenshot