Project d'AL : framework de jeux 2D

guillaume verdugo et ousmane pape

January 2016

Contents

1 Introduction

Dans le cadre du cours d'Architecture logicielle nous devions créer un jeu 2D utilisant le framework présenté avec le jeu pacman. Nous avions comme consigne de ne pas modifier la base du framework. Cette consigne impliquait que nous devions donc en assumer les eventuelles restrictions qu'il comportait.

2 Présentation

2.1 Résumé

Nous avons donc choisi d'implémenter pour ce projet un labyrinthe 2D dans lequel, notre personnage doit braver de multiples dangers pour arrivé à la fin de ce niveau. Comme danger il y a des monstres plus ou moins puissant et des boules de feu qui rebondisent entre elles et donc peuvent être imprévisible. Notre personnage a la possibilité de prendre des bonus comme par exemple des vies supplémentaires et a aussi la possibilités de prendre des bonus mystères (qui peuvent être utile pour le personnage ou a contrario peut lui barrer la route) qui se situent un peu partout sur la carte.

2.2 Entités

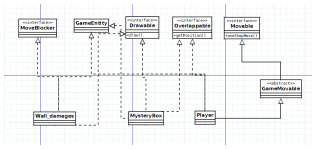

Figure 1: Les trois différents types d'entitées

L'image ci-dessus présente les trois types d'entités présentent dans notre labyrinthe 2D. Nous avons choisi de ne pas afficher l'ensemble des entités ajoutées sur l'image par soucis de compréhension. Toutes ces entitées implementent les interfaces Drawable et GameEntity qui leurs permet d'être inclus a l'univers du niveau courant. Ensuite vient le tour des spécificitées selon nos besoins. Il y a deux types de spécificitées qui sont:

moveBlocker : Ce sont des entitées figés et qui pourront intéragir avec des entitées qui héritent de GameMovable.

Overlappable : Permet a deux entitées de ce même type de se superposées. Il y a certaines entitées dans cette catégorie qui héritent de GameMovable pour leurs permettent de ce déplacer sur la map grâce à un speedVector.

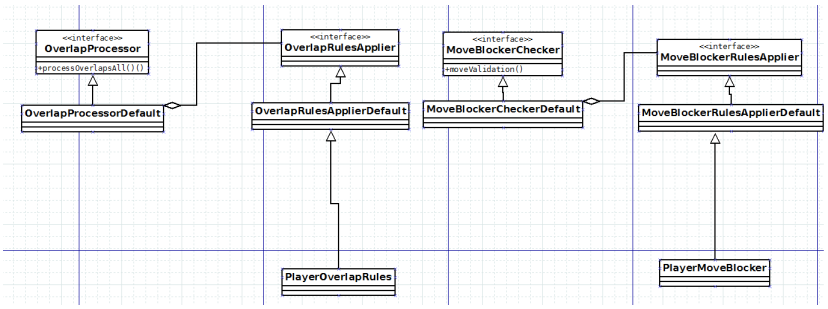

Figure 2: règles et leurs connexions

2.3 Règles

Dans notre labyrinthe 2D nous avons défini deux types de règles concernant les entités qui sont:

PlayerOverlapRules : Cette classe définis les règles de chevauchement des éléments de type Overlappable. Elle hérite de OverlapRulesApplierDefaultImpl qui va permettre d'appliquer une règle de type overlapRule sur deux objets spécifiques en utilisant l'introspection. Par exemple si un objet de type fireball rencontre un autre objet de type fireball notre règle va regarder si il existe une règle overlapRule(Fireball obj, Fireball obj2).

PlayerMoveBlocker : Dans cette classe on va mettre l'ensemble des règles à appliquer quand de tels éléments vont rencontrer des éléments GameMovable. Cette classe utilise également l'introspection pour appliquer une méthode de type moveBlockerRule à un élement GameMovable.

2.4 Drivers et stratégies

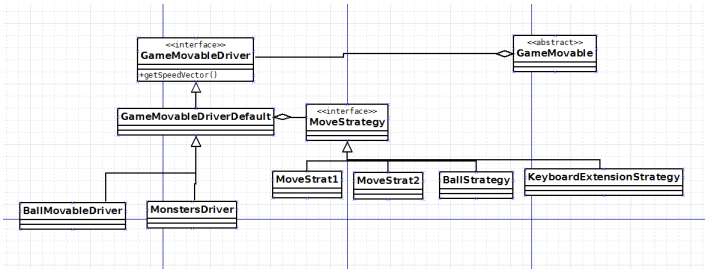

Figure 3: règles et leurs connexions

Le driver est l'élément essentiel pour une entité Overlappable de type GameMovable puisqu'il permet de récupérer un SpeedVector qui va permettre par la suite l'effet de mouvement lors de chaque affichage. Pendant la boucle du jeu

(méthode run de GameLevelDefaultImpl) le système fait appel à une méthode nommée allOneStepMoves qui va elle même faire appelée la méthode OneStepMove des entitées overlappables movable. Cette méthode OneStepMove va cherché le speedVector du driver que lui même va chercher dans notre stratégie pour notre entité. Ce driver va ensuite réaliser une série de traitement si besoin pour vérifier que le speedVector est valide. Ceci ce fait grace à la méthode moveValidation de MoveBlocker qui vérifie si la translation du personnage est possible ou non par rapport aux moveBlockers présents dans le niveau. Par exemple pour nos boules de feu si le moveValidation n'est pas valide alors nous allons chercher dans un tableau de speedVector un speedVector choisi aléatoirement qui est valide et qu'on appliquera à notre stratégie. En faisant cela on réalise l'effets de rebonds avec une direction aléatoire valide sur les murs qui sont des entités moveBlocker.

2.5 Game et level

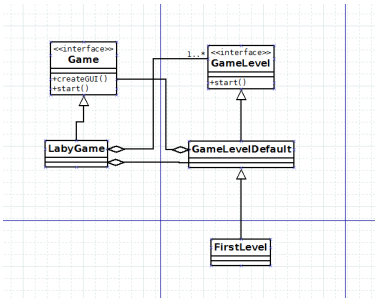

Figure 4: game et niveau

Commencons par notre classe qui définira la base de l'interface utilisateur. Cette classe s'appelle LabyGame qui hérite de Game. On va définir une liste de level , la taille de la frame et la barre de menu. On va pouvoir ajouter différentes valeurs dans le menu comme les valeurs par défault life, score, level et aussi les événements propres à chaque item du menu. On va pouvoir égalements mettre observables certaines valeurs pour pouvoir les modifiés tout au long du jeu (life, score par exemple).

Ensuite viens le tour de notre niveau. Cette classe va utiliser:

OverlapProcessor : Dans cette classe on va définir quels éléments sont movable ou pas. Elle va posséder également des règles de superposition donc dans notre cas PlayerOverlapRules et c'est elle qui vérifie si la superposition est possible.

MoveBlockerChecker : Elle fait la même chose qu'overlapProcessor pour les éléments MoveBlocker et vérifie si le mouvement est possible avec moveValidation.

GameUniverse : On va définir l'universe grâce a OverlapProcessor et MoveBlockerChecker. C'est ici qu'on va ajouter les entités qui se mettront soit dans les listes d'overlapProcessor ou MoveBlockerChecker selon leurs type. C'est également ici qu'on lance la fonction oneStepMove des Movable et que ce lance la verification de la possible superposition des éléments.

GameUniverseViewPort : Elle affiche les éléments dans notre frame de jeu et également le fond.

3 Justifications choix architectures

4 Problèmes

Nous avons rencontrés quelques difficultées pendant ce projet. Pour commencé étant donné qu'on ne devais pas touché au framework de base nous avons été limités par les fonctions disponibles des interfaces. Beaucoup de fonctions sont innacessible et on doit trouver un moyen beaucoup moins évident pour réaliser l'opération voulu. Prenons comme exemple la classe GameUniverseViewPortDefaultImpl, elle possède une méthode setBackground qui est inaccessible et qui oblige donc à utilisé le background de la classe qui est initialisé en utilisant la méthode backgroundImage() comportant le chemin du fichier. Du coup nous avons du étendre cette classe pour redefinir la méthode backgroundImage() avec notre chemin.