

Compte-Rendu de Projet Logiciel Transversal

Zhang Wenyuan - Denamganai Kevin

1^{er} octobre 2015

TABLE DES MATIÈRES

i	OBJECTIF	4
1	PRÉSENTATION GÉNÉRALE	5
1.1	Inspiration	5
1.2	Règles du jeu	5
ii	DESCRIPTION ET CONCEPTION DES ÉTATS	6
2	DESCRIPTION DES ÉTATS	7
2.1	Éléments fixes	7
2.2	Éléments mobiles	7
2.3	Etat général	8
3	CONCEPTION LOGICIELLE	9
3.1	Composition d'élément	9
3.2	Fabrique d'élément	9
3.3	Environnement	9
3.4	Attributs de la classe <i>IElementMobile</i> en fonction du moteur physique	9

TABLE DES FIGURES

FIGURE 1	Illustration de l'animé Robotics;Notes et du jeu <i>KillBallad</i> .	5
FIGURE 2	Diagramme de classe des états	10

Première partie

OBJECTIF

PRÉSENTATION GÉNÉRALE

1.1 INSPIRATION

L'objectif de ce projet est la réalisation d'un jeu proche du jeu fictif *KillBallad* présenté dans l'animé *Robotics;Notes* (cf. Figure 1). Ce jeu a pour particularité de faire intervenir un gameplay très proche de la commande d'un robot humanoïde. Ce sera un aspect central dans notre projet sans pour autant oublier l'aspect tour par tour.

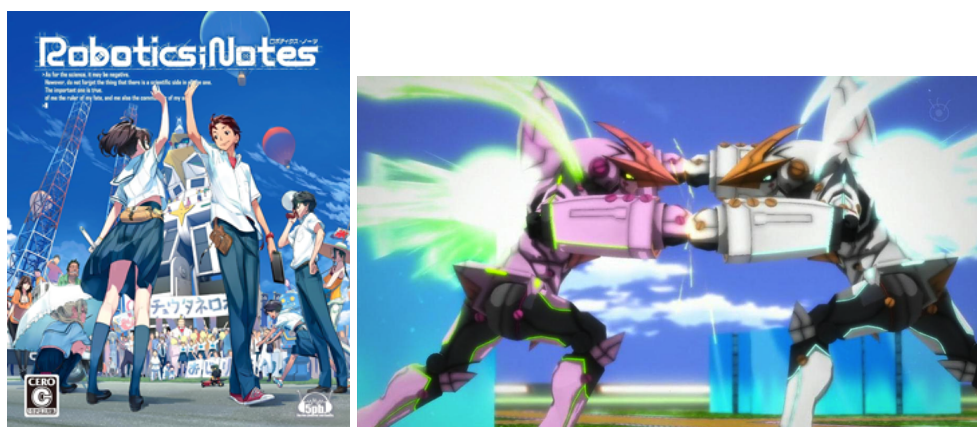


FIGURE 1 – Illustration de l'animé *Robotics;Notes* et du jeu *KillBallad*.

1.2 RÈGLES DU JEU

Initialement, deux modes de jeu sont prévus :

- Mode **Course** : le joueur doit amener son personnage/robot jusqu'à la ligne d'arrivée le plus rapidement possible (en terme de nombre de tour joué), en évitant les obstacles sur sa route. Il pourra y avoir présence ou non d'un ou plusieurs adversaires.
- Mode **Combat** : Cela consiste en un combat à la façon des Sumotoris, le joueur faisant face à un adversaire, son but est de contrôler son personnage/robot de sorte à appliquer des forces qui vont déséquilibrer son opposant et le faire tomber. La chute donnant le perdant.

Toutefois, dans le but de ne pas trop compliquer le projet, nous commencerons par nous concentrer sur le Mode **Course** qui présente le moins de difficultés.

Deuxième partie

DESCRIPTION ET CONCEPTION DES ÉTATS

DESCRIPTION DES ÉTATS

Un état du jeu peut se décomposer autour d'éléments fixes et d'éléments mobiles qui, au départ, sont tous des éléments comportant les attributs suivants :

- la Pose : $x \in \mathbb{SE}(3)$, l'espace spécial euclidien de dimension 3, qui décrit entièrement la position et l'attitude de l'objet.
- un identifiant qui permet de déterminer le type (de classe) de l'élément.

2.1 ÉLÉMENTS FIXES

L'environnement est composé d'un sol et d'obstacle à franchir. Le milieu de taille limité, prédéfini selon la course choisie. Parmi les éléments fixes de l'environnement, on compte :

- les **Obstacles** : Il s'agit d'éléments de tailles variable franchissable ou non, selon la taille. Par exemple, le sol est un obstacle infranchissable car de taille maximale.
- les **Orbes Bonus** : qui peuvent donner des capacités telles que le fait de pouvoir briser/détruire des obstacles franchissables pendant un temps déterminé ou un nombre de coups possibles.

2.2 ÉLÉMENTS MOBILES

Au sein de l'environnement, en ne considérant toujours que le mode **Course**, les éléments mobiles sont les personnages/robots. Plus exactement, on va définir les personnages/robots comme une composition d'éléments mobiles de bases chaînés entres eux qui contiendront donc des attributs supplémentaires, nécessaire au bon fonctionnement du moteur physique qui devra gérer les évolutions de ces éléments mobiles, telles que :

- un torseur cinématique : une résultante $\vec{\Omega} \in \mathbb{R}^3$, vecteur vitesse de rotation et un vecteur vitesse d'un point $P \in S$ le solide/élément mobile, $\vec{v}_P \in \mathbb{R}^3$
- un torseur dynamique : une résultante $\vec{A} \in \mathbb{R}^3$, vecteur quantité d'accélération et un moment dynamique en un point $P \in S$ le solide/élément mobile, $\vec{\delta}_P \in \mathbb{R}^3$
- la liste (map) des liaisons cinématiques de l'élément avec d'autres éléments.
- ...

Pour le moment, comme on le verra dans le diagramme de classe, les choix à l'égard du moteur physique n'étant pas encore arrêtés, cette partie du moteur d'états n'est pas encore décidée.

En ce qui concerne le bonus, notre personnage/robot va pouvoir acquérir des capacités pendant une durée déterminée en nombre de tour parmi les suivantes :

- devenir plus grand : et ainsi se mouvoir plus facilement, par exemple car des obstacles infranchissable par le passé à cause de leur taille deviendraient maintenant franchissable.
- obtenir la capacité à briser/détruire les obstacles.
- ralentir l'adversaire pendant un certain nombre de tour.

2.3 ETAT GÉNÉRAL

A l'ensemble des éléments fixes et mobiles, on rajoute des attributs tels que :

- Le compteur d'orbes bonus encore présentes dans l'environnement.
- Une horloge qui compte le temps qui s'est écoulé afin que les intégrations des mouvements de chaque solide les uns par rapport aux autres soient toutes synchrones.
- Une vitesse à laquelle l'horloge du jeu se met à jour par rapport au nombre de tour et/ou au temps réel.

CONCEPTION LOGICIELLE

Le diagramme des classes pour les états est présenté dans la *Figure 2*, dont nous pouvons mettre en évidence les particularités suivants :

3.1 COMPOSITION D'ÉLÉMENT

Les éléments mobiles, étant uniquement les personnages/robots, il serait logique d'avoir une gestion des parties de chaque personnage/robot sous la forme d'une composition. C'est la raison pour laquelle le *pattern compos* est utilisé.

3.2 FABRIQUE D'ÉLÉMENT

Les éléments fixes, dont nous avons besoin de créer un grand nombre afin de construire les pistes sur lesquels les courses auront lieu(en se référant au mode de course), seront générés à l'aide du *pattern factory* afin de répondre efficacement à ces besoins de génération en grand nombre..

3.3 ENVIRONNEMENT

La classe **Environnement** est la classe par laquelle on peut gérer facilement les éléments. Elle nous permet de supprimer et ajouter certains éléments pendant le déroulement du jeu et en même temps d'accéder facilement à ces éléments dans le but de mettre à jour leurs attributs le cas échéant.

3.4 ATTRIBUTS DE LA CLASSE *ielementmobile* EN FONCTION DU MOTEUR PHYSIQUE

Ne sachant pas encore tous les détails concernant le moteur physique, il n'est pas encore possible de savoir quelles seront les attributs nécessaires pour la gestion des mouvements des éléments mobiles. De plus ce moteur physique est susceptible de changer, dès lors de le *pattern strategy* rendra cette évolution plus facile au niveau des états.

3.4 ATTRIBUTS DE LA CLASSE *ielementmobile* EN FONCTION DU MOTEUR PHYSIQUE

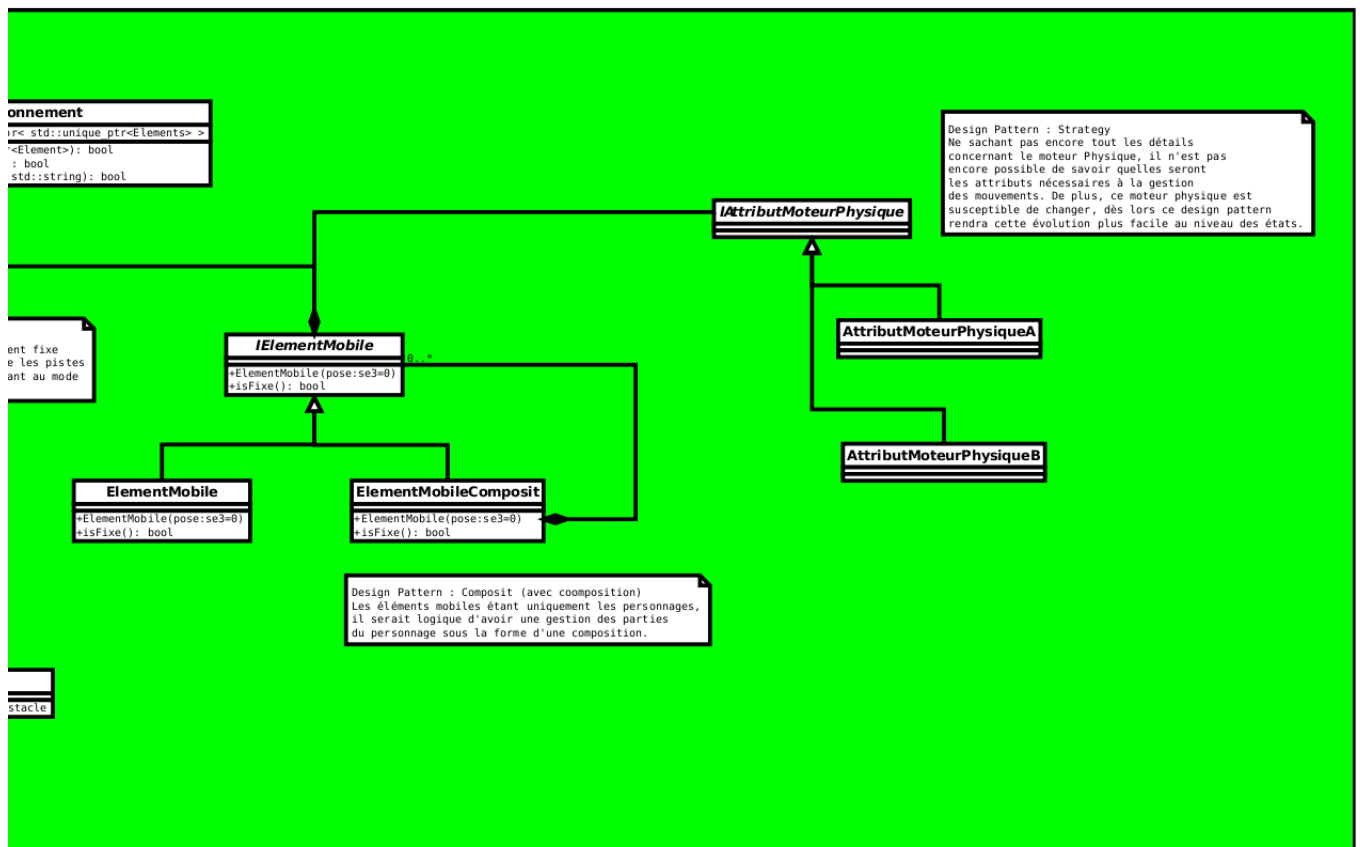
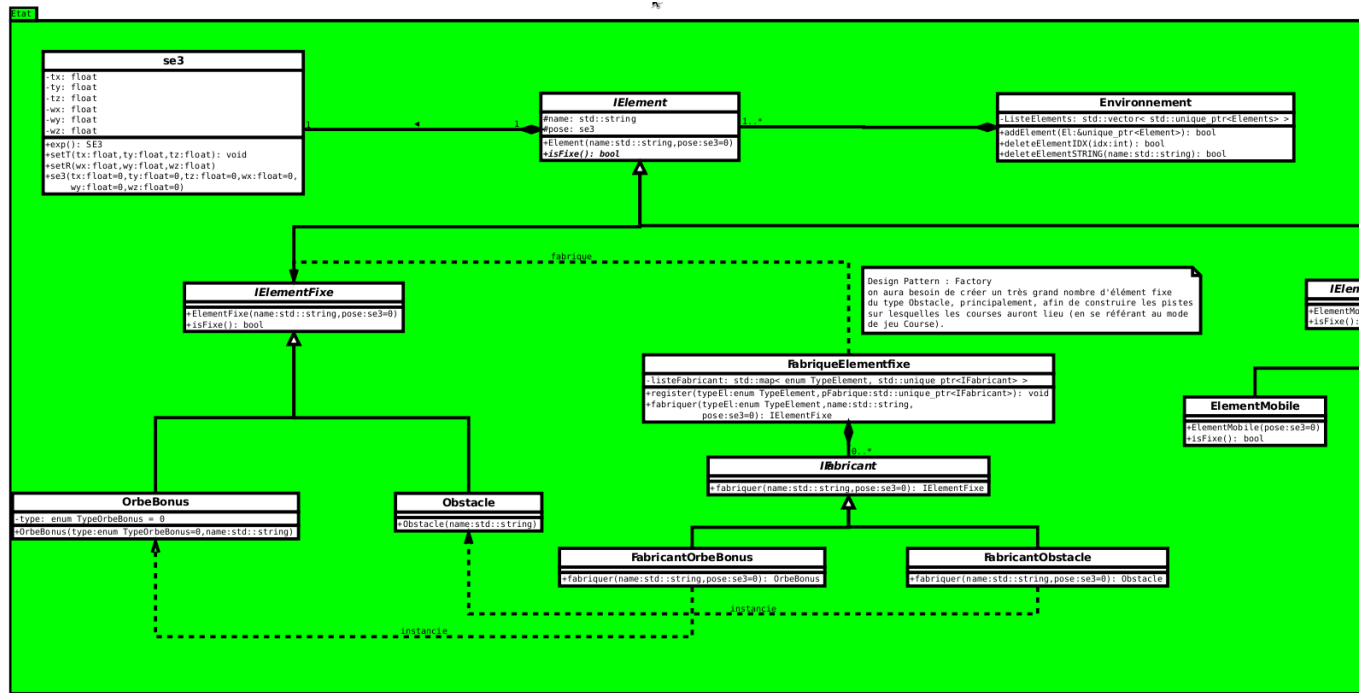


FIGURE 2 – Diagramme de classe des états

BIBLIOGRAPHIE
