|  |
| --- |
| ***YouRobot*** |
| *Rapport de projet* |

|  |  |  |  |
| --- | --- | --- | --- |
| **Identification** | | | |
| Client | **M. FORAX Remy** | | |
| Réf. Projet | **MLV00003** | | |
| Réf. Doc | **FO01005-ETDG-A**  ***Merci de rappeler cette référence dans toute correspondance.*** | Rév. | A |
| Date | **29/11/11** | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Validation** | | | |
| Action | Date | Nom | Visa |
| Rédaction | 29/11/11 | Girard Damien, Joan Goyeau |  |
| Vérification | 30/11/11 | CLOUET Emmanuelle |  |
|  |  |  |  |
|  |  |  |  |

|  |  |  |
| --- | --- | --- |
| **Diffusion** | | |
| Date | Destinataire | Société |
| 30/11/11 | M. FORAX Remy | Université Marne la Vallée |
|  |  |  |



Sommaire

**Table des matières :**

[1 Présentation générale du sujet 3](#_Toc310454315)

[2 Technologies utilisées et moyens 3](#_Toc310454316)

[2.1 Environnement de développement 3](#_Toc310454317)

[2.2 Contrôle du code source et travail collaboratif 3](#_Toc310454318)

[2.3 Diagramme UML 3](#_Toc310454319)

[3 Déroulement du projet 4](#_Toc310454320)

[4 Diagrammes de cas d’utilisations 5](#_Toc310454321)

[4.1 Les joueurs 5](#_Toc310454322)

[4.2 Les éléments et le terrain de jeux 6](#_Toc310454323)

[4.3 Conclusion sur les diagrammes de classes 6](#_Toc310454324)

[5 Diagrammes de classes 7](#_Toc310454325)

[1. Diagramme complet de l’application 7](#_Toc310454326)

[5.1 Découpage de l’application 8](#_Toc310454327)

[5.2 Package YouRobot 9](#_Toc310454328)

[5.3 Package elements 10](#_Toc310454329)

[5.4 La classe abstraite Element 11](#_Toc310454330)

[5.5 Packages Area, Robot et Wall 12](#_Toc310454331)

[5.6 Les bonus 13](#_Toc310454332)

[6 Conclusion 14](#_Toc310454333)

**Table des illustrations :**

[Figure 1 Diagramme des cas d’utilisations des joueurs 5](#_Toc310454334)

[Figure 2 Diagramme des cas d’utilisation du terrain 6](#_Toc310454335)

[Figure 3 Diagramme de classe complet 7](#_Toc310454336)

[Figure 4 Packages du jeu 8](#_Toc310454337)

[Figure 5 Classe contenue dans le package YouRobot 9](#_Toc310454338)

[Figure 6 Contenu du package elements 10](#_Toc310454339)

[Figure 7 La classe Element 11](#_Toc310454340)

[Figure 8 Package Area 12](#_Toc310454341)

[Figure 9 Package Robot 12](#_Toc310454342)

[Figure 10 Package wall 13](#_Toc310454343)

[Figure 11 Package bonus 13](#_Toc310454344)

[Figure 12 Ecran d’accueil de YouRobot 14](#_Toc310454345)

# Présentation générale du sujet

L’objectif de ce projet est la création d’un jeu vidéo en Java, l’aspect pédagogique visé derrière sont les suivants :

* Apprendre à utiliser des bibliothèques Java très peux documentée (Box2D et Zen)
* Réaliser une application complète en java
* Mettre en pratique les notions vues en POO et Java
* Pouvoir créer des jeux 2D grâce à Box2D
* Réaliser un projet dans un temps très court : gestion du stress

Le sujet donné par M. FORAX Remy laissait certain point à l’appréciation personnel des développeurs. Nous avons suivie au maximum le sujet donné.

Le sujet est disponible à l’adresse suivante :

<http://igm.univ-mlv.fr/ens/IR/IR2/2011-2012/Java_Avance/project.php>

# Technologies utilisées et moyens

## Environnement de développement

Nous avons utilisée l’environnement de développement NetBeans. Cet IDE [[1]](#footnote-1)est très puissants même s’il ne dispose pas de toute les fonctionnalités offertes par son principal concurrent Eclipse, il a l’avantage d’être très simple à utiliser, ressemble à Visual Studio (Damien Girard programme en C# avec Visual Studio en entreprise).

## Contrôle du code source et travail collaboratif



Le contrôle du code source a été effectué sur Google Code.

<http://code.google.com/p/yourobot>

Nous avons utilisé l’interface Subversion (SVN) pour récupérer et commit notre travail sur Google code.

Netbeans dispose d’un plugin (déjà intégré) permettant d’utiliser facilement Subversion, tout a été fait à partir de l’IDE.

## Diagramme UML

Nous avons utilisé le logiciel Altova UModel pour faire les diagrammes UML de l’application.

Altova UModel ne supporte pas encore Java 7 c’est pourquoi nous avons codé le projet en style Java 6. Néanmoins nous avons utilisé la bibliothèque JDK 7, notamment « Objects.RequireNonNull() ».

# Déroulement du projet

Le projet s’est déroulé selon les phases suivantes :

* Terminer avant tout le projet d’algorithmie
* Lire et appréhender le cahier des charges
* Réalisation des diagrammes de cas d’utilisations
  + Cela donne la première esquisse de l’organisation du code
* Réalisation des diagrammes de classes
  + Et génération par Altova UModel du squelette de l’application.
* Programmation de la partie graphique et affichage
* Apprentissage de JBox2D (et Box2D)
* Implantation dans le projet de JBox2D
* Tests de la partie
* Programmation des menus
* Design et mise en place des niveaux
* Réalisation du rapport

Le projet a été réalisé dans un temps très court (2 semaines complètes). L’analyse complète de l’application et les diagrammes UML ont permis une programmation très rapide et efficace.

# Diagrammes de cas d’utilisations

## Les joueurs

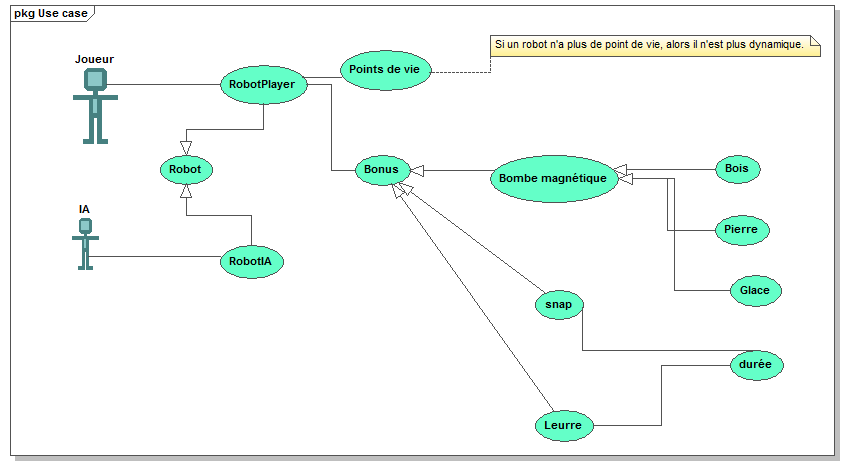


Figure 1 Diagramme des cas d’utilisations des joueurs

Après la lecture du sujet, le premier diagramme à avoir été réalisé a été le diagramme des cas d’utilisations des joueurs.

Nous voyons donc un Joueur et une IA, qui contrôle un robot. Un robot à des points de vue peut avoir des bonus.

Les bonus ont un type (Bois, Pierre, glace) et/ou une durée.

## Les éléments et le terrain de jeux

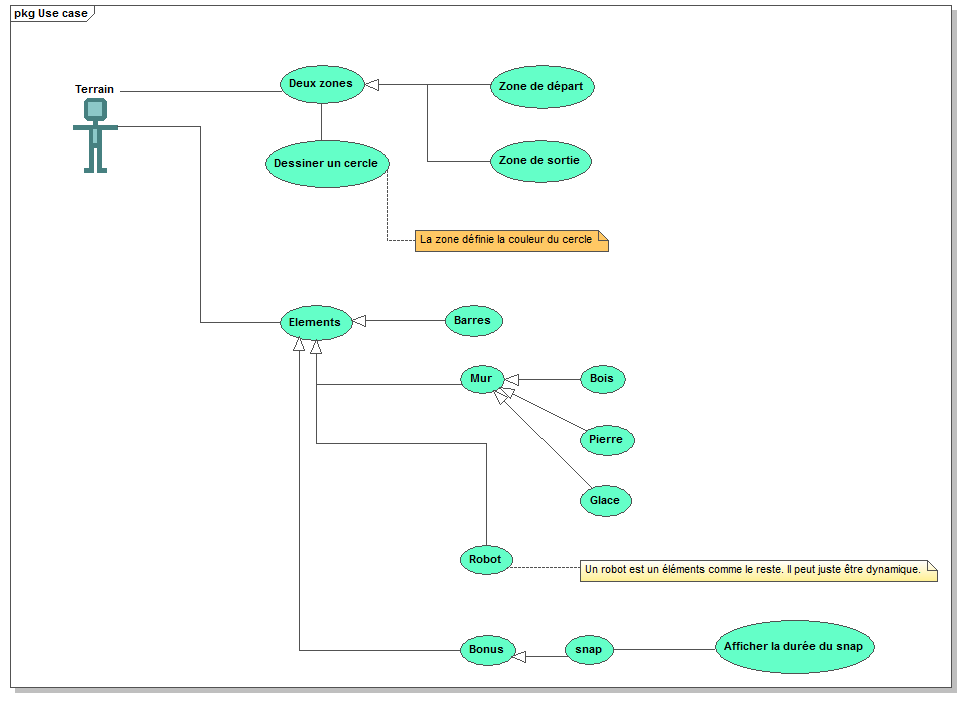


Figure 2 Diagramme des cas d’utilisation du terrain

Le terrain de jeux dispose de plusieurs zones (deux de départ et une de sortie).

Les éléments du jeu sont les suivants :

* Des barres (ne peuvent bouger)
* Des murs (peuvent bouger à cause d’un bonus)
* Des robots
* Des bonus

## Conclusion sur les diagrammes de classes

Grâce au diagramme de classe, nous avons pu formaliser et mettre au clair nos esprits sur ce que le projet devait faire et comment nous devions le faire.

L’étape suivante a été la réalisation des diagrammes de classe.

# Diagrammes de classes

## Diagramme complet de l’application

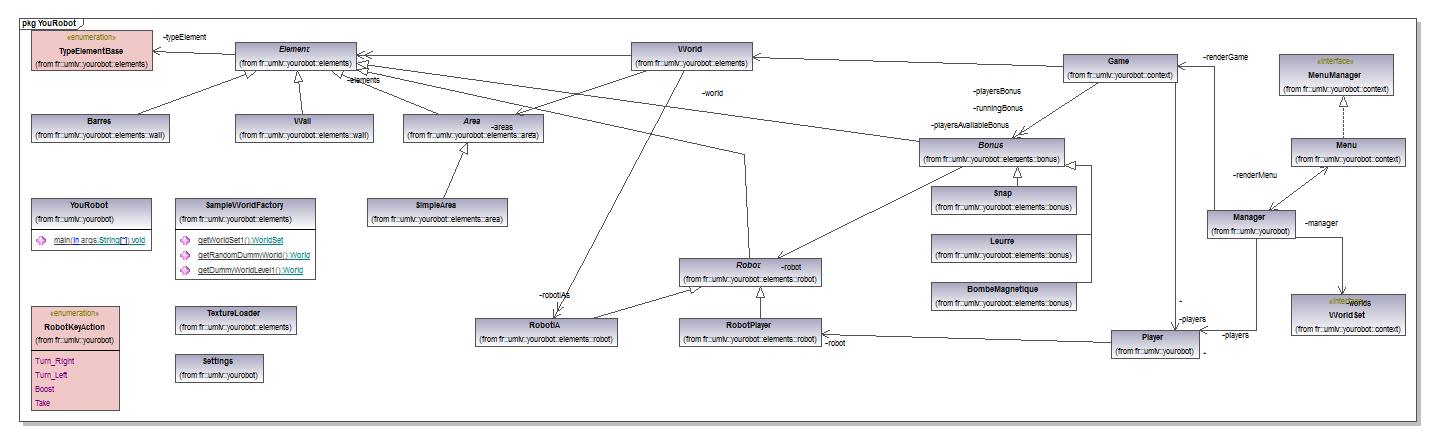


Figure 3 Diagramme de classe complet

Voici le diagramme de classe complet de l’application. Comme vous pouvez le voir l’application est fortement découpé, mais en regardant de plus près le découpage est logique.

Ce diagramme dit : Un monde contient des éléments et des zones, une partie (Game) est composée d’un monde et de bonus et possède un ou deux joueurs humains, un joueur humain est associé à un robot (robotPlayer).

## Découpage de l’application

Nous avons réalisé de nombreux packages pour notre application, l’objectif étant clairement de spécialiser chaque partie du jeu.

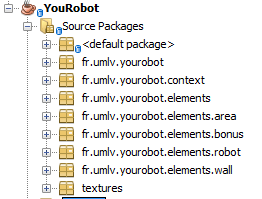


Figure 4 Packages du jeu

Explications des différents packages :

* YouRobot : Paramètres de l’application, classes vraiment générales
* Context : Eléments étant liés au menus et déroulement du jeu
* Elements : Contient la classe abstraite « Element », le monde ainsi que le loader de texture
* Elements.area : Contient les zones de départ et de fin
* Elements.bonus : Contient les différents bonus du jeu
* Elements.robot : Contient les robots
* Elements.wall : Contient les murs et barres

## Package YouRobot

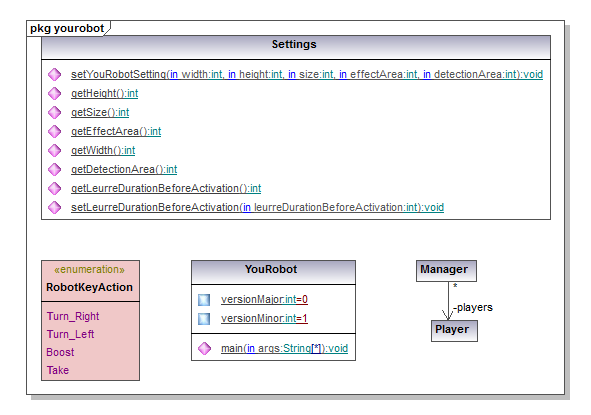


Figure 5 Classe contenue dans le package YouRobot

L’application dispose de paramètres globaux (classe Settings), même si cela n’est pas optimal cela convient parfaitement pour notre cahier des charges, car rarement ces paramètres sont modifiés.

Aussi nous voyons la classe Manager qui gère le comportement de l’application (menu et déroulement du jeu). Cette classe est associée à des Player.

En fait cela est fait comme ça car dans le futur il se peut que nous voulions charger les configurations des joueurs depuis des fichiers de configuration. Donc au démarrage on crée des joueurs et un manager.

## Package elements

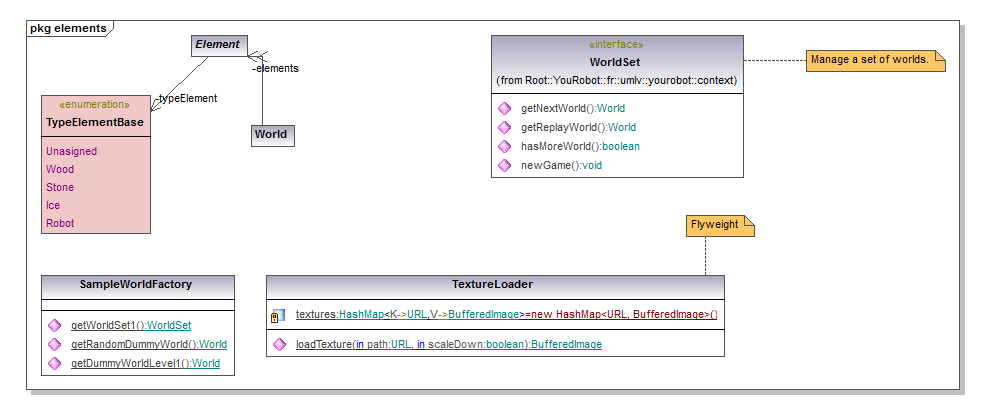


Figure 6 Contenu du package elements

Dans le package elements nous pouvons voir la classe abstraite Element. Cette classe contient le comportement par défaut de n’importe quel élément du jeu.

TextureLoader est une classe implantant le design pattern FlyWeight, il permet de ne charger qu’une seule fois en mémoire une texture.

WorldSet est un ensemble de monde, il est utilisé par la classe Manager pour savoir quel niveau doit-on jouer.

World représente un monde et contient donc des éléments.

Chaque élément dispose aussi d’un type, ce type est utilisé par la suite pour déterminer la puissance des bonus.

## La classe abstraite Element

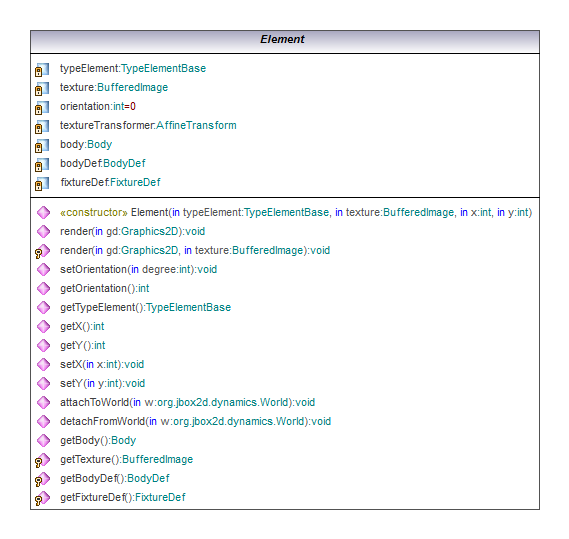


Figure 7 La classe Element

La classe élément est la brique principale de YouRobot. Cette classe permet et effectue les opérations suivantes :

* Dessiner une texture, supporte l’orientation.
* Obtenir et définir la position de la texture.
* S’attacher et se détacher d’un monde (classe World). (attachToWorld et detachFromWorld)
  + L’attache d’un élément à un monde est obligatoire et a été fait comme cela car un Body de Box2D est créé à partir dans World de JBox2D.

Le code d’attache est le suivant :

// Associating this object with the body.

// This is made here and not in the constructor to prevent a constructor this leak.

this.bodyDef.userData = (Object) this;

body = w.createBody(bodyDef); // Creation du body JBox2D.

body.createFixture(fixtureDef); // Association au monde JBox2D.

setOrientation(orientation); // Force la mise à jour de la rotation.

## Packages Area, Robot et Wall

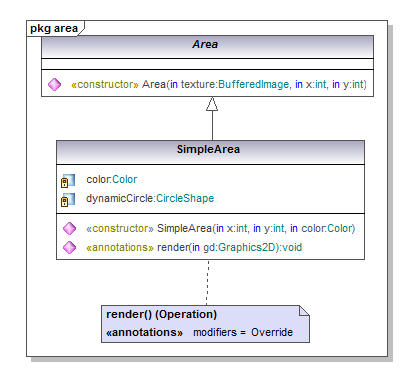


Figure 8 Package Area

Le package area contient une classe abstraite qui définit une zone et SimpleArea.

SimpleArea dessine simplement un cercle unicolor.

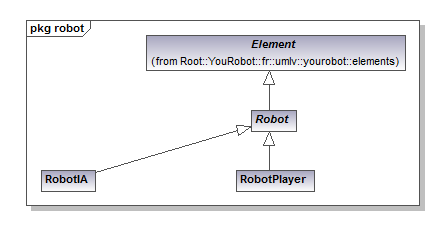


Figure 9 Package Robot

Un robot est un élément et il y a deux type de robot : RobotIA et RobotPlayer.

RobotPlayer est joué par un humain, RobotIA est une IA.

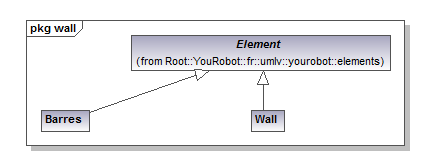


Figure 10 Package wall

Il y a deux types de murs dans le jeu, les statiques (Barres) et les dynamiques (Wall).

## Les bonus

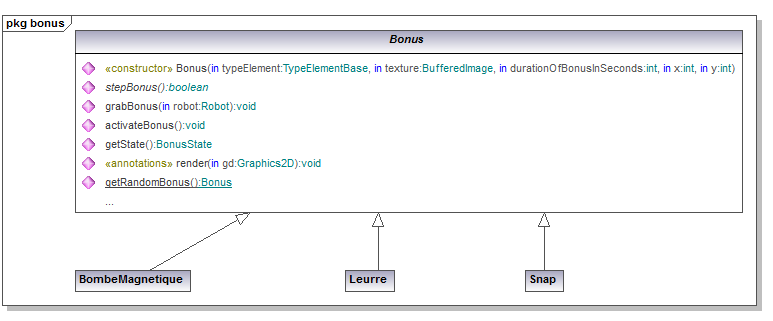


Figure 11 Package bonus

Les bonus fonctionnent comme cela dans notre YouRobot :

* Ils sont placés et géré par le jeu en cours (classe Game)
* Quand un joueur attrape un bonus, le bonus est associé à un Robot (grabBonus)
* Le joueur peut ensuite activer le bonus avec activateBonus()
* A chaque itération dans le jeu (par Game), la méthode stepBonus est appelée
  + Cette méthode permet au bonus d’appliquer son effet « bonus ».

# Conclusion

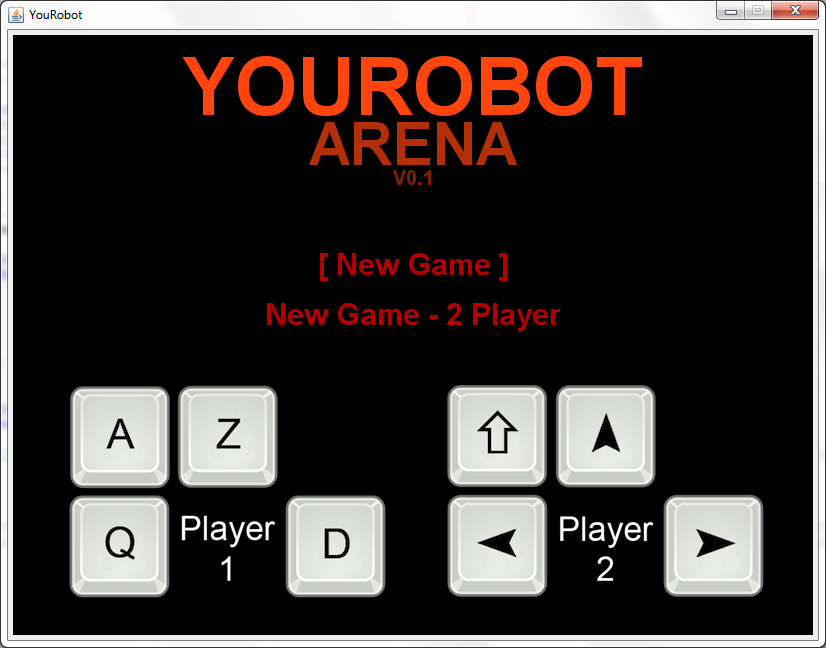


Figure 12 Ecran d’accueil de YouRobot

YouRobot a été réalisé en 2 semaines en sacrifiant toute vie sociale, créant un stress intense mais néanmoins bien géré. Le jeu est fonctionnel, même si il peut recevoir de nombreuses améliorations.

Nous avons essayé de coder le plus proprement possible et le plus dans l’esprit Java orienté objet, bien que cela n’a pas été toujours facile car de vieux automatismes en C#/C ressurgissais de temps en temps.

YouRobot contient 6895 lignes soit 226 001 caractères (incluant la documentation), c’est un projet assez conséquent. Mais le résultat est là, le jeu est sympathique à jouer.

Le projet est sous licence GPLv3 et est disponible à l’adresse suivante :

<http://code.google.com/p/yourobot>

La javadoc est disponible à l’adresse suivante :

<http://yourobot.googlecode.com/svn/trunk/javadoc/index.html>

1. Integrated Development Environment [↑](#footnote-ref-1)