PROGRAMMTION AVANCEE

RAPPORT DE PROJET

MASTER 1 – MIAGE

Monsieur Fabrice HUET

Executive Summary

The project :

The goal of this project is to build a virtual fighting game of the « RobotWar » type.

In a battle arena in 2D, seen from above, robots clash, managed by a relatively basic AI. The behavior and the graphics of robots is decided by plugins. A robot is active if its life has not reached 0 and the winner is the last active robot. Each robot is associated with a quantity of energy and each action consumes a part of it. The energy goes up regularly until it reaches the maximum value.

Le projet :

Le but de ce projet est de construire un jeu de combat virtuel de type « RobotWar ».

Dans une arène de combat en 2D, vue de dessus, des robots s'affrontent, gérés par une IA relativement basique. Le comportement ainsi que le graphisme des robots est décidé par des plugins. Un robot est actif tant que sa vie n'a pas atteint 0 et le gagnant est le dernier robot actif. À chaque robot est associé une quantité d'énergie et chaque action consomme une partie de celle-ci. L'énergie remonte régulièrement tant qu'elle n'a pas atteint la valeur maximale.

Sommaire

PARTIE 1. PRÉSENTATION DES MEMBRES DU GROUPE 4
DATAO Chen 4
FONTAINE Gaël 4
ROMAN Geoffrey 5
PARTIE 2. PROCEDURE A SUIVRE POUR TESTER LE PROJET 5
PARTIE 3. CALENDRIER DES GRANDES ETAPES 6
PARTIE 4. REALISATION DES CINQ POINTS DU PROJET 8
Fonctionnalité 8
Mécanisme de gestion de plugins et de chargement dynamique 8
Persistance (sauvegarde de l'état des plugins) 8
Modularité et dépendances 9
Documentation / gestion du projet 9
PARTIE 5. PROCEDURE POUR CREER UN NOUVEAU PLUGIN 9
PARTIE 6. EXEMPLES DE PLUGINS 10
1. Plugin d'attaque 10
2. Plugin de déplacement 10
3. Plugin graphisme 11

PARTIE 1. PRÉSENTATION DES MEMBRES DU GROUPE

DATAO Chen

Au cours de ce projet, j'ai réalisé différentes taches dans plusieurs domaines distinct. Tout d'abord je me suis occupé de la classe centrale servant aux robots. Par la suite, j'ai créé un plugin graphique visant à générer un aspect avancée du robot. Enfin, j'ai mis au monde le plugin d'attaque à distance dont le but est qu'un robot puisse en attaquer un autre à longue portée.

En conclusion, je dirai que le projet ma apporter énormément dans la partie plus « informatique » avec le langage Java (utilisation de projet maven, …). Mais également dans la partie gestion de projet et gestion d'équipe où j'ai pu collaborer avec mes coéquipiers.

FONTAINE Gaël

Mon travail au sein du groupe et dans le projet en général est divisé en plusieurs parties. Premièrement, la structure et la découpe du projet « maven » en 3 modules distinct (moteur, annotation et plugin). J'ai également pu rédiger 1 plugin d'attaque et 1 plugin de graphisme sur le projet, « *AttaqueSimple* » qui permet à deux Robot de s'affronter et « *NomRobot* » qui place le nom des robots au-dessus de ces derniers en cours de partie.

De plus, mon rôle aura été de créer la classe « Launch » qui permet de lancer une partie et de dérouler un scénario quant à la mort des robots. J'ai également participé au moteur du jeu avec la classe « *FrameWithMenu* » que j'ai modifié plusieurs fois afin d'optimiser le fonctionnement du jeu.

Enfin, la dernière partie concrète aura été de rédiger les tests unitaires de certaines classes permettant de tester les classes centrales du jeu. Pour conclure, le projet m'a permis de mettre en pratique toutes les connaissances accumulées au cours du semestre de M1 MIAGE et de m'améliorer dans le travail d'équipe.

ROMAN Geoffrey

Mon travail durant ce projet a été, de créer un « classLoader » personnalisé permettant de charger des classes dans un dossier spécifique, ou dans un JAR. J'ai aussi créé une classe Repository, permettant de charger toutes les classes dans le projet, ayant une annotation spécifique à un des trois types de plugins.

De plus, je me suis chargé de faire la classe FrameWithMenu, qui permet d'afficher la fenêtre du jeu avec les différents robots qui se déplacent, attaque... La fenêtre est composée d'un menu permettant soit de sauvegarder/charger une partie, soit d'utiliser un plugin de type graphique.

Je me suis aussi occupé de réaliser la classe ChargementPlugin, qui permet, en passant en paramètre la classe du plugin ainsi que la FrameWithMenu dans laquelle le plugin doit être chargé, d'appeler la méthode d'action du plugin (« *draw* », « *deplacement* » ou « *attaque* »)

J'ai aussi fait le système de sauvegarde/chargement de partie, grâce à la partie sérialisation vu en cours.

En ce qui concerne la partie plugin, ma tâche a été de faire le plugin de déplacement simple

PARTIE 2. PROCEDURE A SUIVRE POUR TESTER LE PROJET

Une fois le projet récupéré, il vous suffit de placer un terminal ayant pour position la racine du projet. Par la suite vous suffira d'exécuter cette commande :

java -jar moteur/launch.jar

PARTIE 3. CALENDRIER DES GRANDES ETAPES

Dans sa globalité le projet de programmation avancée aura duré près de 1 mois et 1 semaines. Plus précisément, le projet a commencé le 4 décembre 2017 et s'est terminé le 12 janvier 2018. Voici les grandes dates par membres :

- CHEN Datao

Date clé	Tâche réalisée
5 janvier 2018	Amélioration de la classe *Robot*
7 janvier 2018	Ajout du plugin graphique compliqué
11 janvier 2018	Soutenance orale

- FONTAINE Gaël

Date clé	Tâche réalisée
13 décembre 2017	Création du repository sur Bitbucket
21 décembre 2017	Mise en place du projet
28 décembre 2017	Mise à jour de la conception du projet (modules maven)
2 janvier 2018	Ajout de la vie pour un robot + Ajout de la classe *Projectile*
4 janvier 2018	Ajout de l'énergie pour un robot
5 janvier 2018	Ajout du plugin d'attaque à courte portée
6 janvier 2018	Mise en place du scénario du jeu + plugin graphique nom du robot

7 janvier 2018	Rédaction des tests unitaire pour le moteur du jeu
11 janvier 2018	Soutenance orale

- ROMAN Geoffrey

Date clé	Tâche réalisée
23 décembre 2017	Ajout de la classe *Robot* + plugin déplacement
28 décembre 2017	Ajout du chargement des plugins + moteur du jeu
4 janvier 2018	Ajout du plugin graphique barre de vie + lancement du jeu
5 janvier 2018	Ajout de la classe *ChargementPlugin*
6 janvier 2018	Ajout du plugin graphique moyen
7 janvier 2018	Implémentation de la mort d'un robot
8 janvier 2018	Ajout du système de sauvegarde et de chargement de la partie
11 janvier 2018	Soutenance orale

PARTIE 4. REALISATION DES CINQ POINTS DU PROJET

Fonctionnalité

- ✓ Concernant les plugins

Au cours de la partie, l'utilisateur peut ou non choisir dans les plugins graphique présent dans l'application. Ces derniers seront chargés instantanément et les changements seront visibles directement par l'utilisateur. Le gros avantages est que la partie n'a aucunement besoin de s'arrêter afin de charger les plugins graphique, les changements se font de manière « responsive ».

De plus, les plugins d'attaques et de déplacement ne seront visibles de l'utilisateur une fois la partie lancé. En effet, c'est au robot de se déplacer et d'attaquer lorsqu'il le souhaite.

- ✓ Sauvegarde et chargement de la partie

Dans une partie, l'utilisateur pourra également sauvegarder la partie en cours en cliquer sur le bouton sauvegarder. Un fichier contenant toutes les informations de la partie sera alors enregistré. En plus, l'utilisateur pet un n'importe quel moment charger ce fichier et retourner à l'état dans lequel il avait sauvegarder précédemment.

Mécanisme de gestion de plugins et de chargement dynamique

C'est la classe Repository qui s'occupe de charger dynamique chaque plugin du projet, elle va dans un premier temps récupérer toutes les classes pour ensuite vérifier si une annotation signifiant que c'est un plugin est présent, si c'est le cas la classe sera stocké dans son ArrayList correspondante (*listePluginsAttaque*, *listePluginsDeplacement*, *listePluginsGraphisme*). Par la suite ces listes seront récupérées dans la classe FrameWithMenu, pour ensuite être utilisé dans la classe Launch, qui va s'occuper d'appeler les méthodes d'actions de chaque plugin à utiliser.

Persistance (sauvegarde de l'état des plugins)

Lors que l'utilisateur souhaite sauvegarder la partie, cela va faire appel à la Sérialisation, qui comme son nom l'indique, va sauvegarder la partie grâce à la sérialisation en écrivant dans un ObjectOutputStream l'ensemble des plugins graphiques utilisés, ainsi que le plugin de déplacement et d'attaque en cours d'utilisation.

Modularité et dépendances

Le module contenant le moteur dépend de JUnit pour effectuer les tests unitaires. Quant à lui, le module avec les annotations ne dépend d'aucun autre module ou de bibliothéque. Par contre, le moteur et les plugins dépendent des annotations.

Documentation / gestion du projet

En ce qui concerne la gestion de projet, toute l'équipe à utiliser les outils présents sur la plateforme Bitbucket ainsi que Sourcetree. Nous avons donc pu, grâce au Trello, créer des taches, les assignés à chaque membre ainsi que leurs données une deadline. Cet outil nous a permis de savoir exactement sur quoi travail chaque membre et ainsi géré au mieux le futur du projet.

De plus, chaque membre avait à sa disposition l'outil Sourcetree permettant d'effectuer toutes les opérations de git avec une interface graphique fiable. Par conséquent nous avions tous un visuel sur l'avancée du projet.

PARTIE 5. PROCEDURE POUR CREER UN NOUVEAU PLUGIN

Afin de créer un nouveau plugin, il vous suffira de suivre ces étapes :

- Création du plugin

Cette étape consiste à créer la base du plugin. Pour cela, le module « plugins » sera le mieux adapté pour répondre à cette exigence. Par la suite, 3 choix s'offre à vous :

- attaque : vous voulez créer une attaque, ou une stratégie de combat ?

- déplacement : vous voulez créer un style de déplacement ou une intelligence à se positionner ?

- graphique : vous voulez améliorer l'aspect du jeu en proposant un nouveau visuel ?

Après libre à vous de choisir le package java qui vous concernera. Un fois choisi, il vous suffira de créer une classe java.

- Développement du plugin

Pour commencer son plugin, il vous faudra placer au-dessus de votre méthode d'action, une annotation relative au package dans lequel vous êtes :

- @Attaque(nom = « nomQueVousVoulez »)

- @Deplacement(nom = « nomQueVousVoulez »)

- @Graphisme(nom = « nomQueVousVoulez »)

Pour information ces annotations serviront par la suite a chargé vos plugins. Pour vous aider, nous allons montrer quelques exemples de plugin existant de base dans l'application.

PARTIE 6. EXEMPLES DE PLUGINS

1. Plugin d'attaque

AttaqueSimple : ce plugin permet aux robots de s'affronter sur une courte distance. Lorsqu'un robot utilise une attaque à courte portée de l'énergie lui est retiré ; Lorsqu'il en a plus, ce dernier ne peut donc plus attaquer. Les dégâts de cette attaque sont variables et peuvent retirer jusqu'à 50% de la vie du robot si elle est réussie.

AttaqueDistante : ce plugin permet aux robots de s'affronter sur une longue distance. Lorsqu'un robot utilise une attaque à longue portée de l'énergie lui est retiré ; Lorsqu'il en a plus, ce dernier ne peut donc plus attaquer. Les dégâts de cette attaque sont moins efficaces que le type d'attaque vu précédemment. A contrario, cette attaque reste tout de même très dangereuse si un robot possède peut de vie !

2. Plugin de déplacement

DeplacementSimple : c'est le système de déplacement par défaut de ce jeu, les robots ont 4 directions possible et le déplacement se fait aléatoirement. Ce système permet aux robots de se marcher dessus et donc de pouvoir s'attaquer au corps à corps !

3. Plugin graphisme

GraphismeSimple, GraphismeMoyen et GraphismeComplique : ces 3 plugins permettent de modifier l'aspect des robots. Comme leurs noms l'indiquent, ces derniers sont plus ou moins précis.

BarreDeVie : ce plugin graphique permet d'afficher la barre de vie d'un robot au-dessus de sa tête Elle est actualisée en direct afin de suivre le combat.

NomRobot : ce plugin ajoute le nom du robot au-dessus de lui, il suit le mouvement de ce dernier afin que l'on sache en permanence qui est qui !