Université Nice Sophia Antipolis

ROBOT WAR 2017

Flavien BOSQUES Guillaume DORE Fabien LOPEZ Mamadou THIAW

M1 MIAGE 1645 route des lucioles

Sommaire

Partie I: Présentation du jeu Robot War

Introduction

Cette année dans le cadre du module de programmation avancée du semestre 1 du Master 1 MIAGE, il a été décidé que le projet porterait sur un jeu de combat de type Robot War, un genre de jeu proposé pour la première dans les années 70 par Silas Warner. Jeu que l'on retrouvera plus tard en 1981 sur Apple II. Le principe est simple, dans une arène en 2D avec un angle de caméra en vue de dessus, des robots s'affrontent. La gestion des robots est gérée par une intelligence artificielle basique.
Nos robots, que ce soit pour le comportement ou la partie graphique seront gérés par des Plugins. Un robot aura donc un nombre de point de vie (100) et sera retiré du jeu lorsque celle-ci atteindra 0. Le gagnant sera le dernier robot en vie dans l'arène. Un robot pourra effectuer un certain nombre d'actions. Celles-ci coûteront à chaque fois de l'énergie. Nos robots auront donc une barre de vie et une barre d'énergie.

Partie II: Méthodologie et Conception

A. Participation des membres du groupe

Fabien Lopez : Réalisation du système d'annotations au niveau du chargement. En effet, dans un premier le chargement des classes fait par Mamadou était basique donc il a ajouté à celui-ci un système de vérification par les annotations pour charger les plugins au programme. De plus, il a participé au développement des plugins et plus précisément le plugin ImageAléatoire qui affecte une image (.png) aux robots de l'arène. Enfin, il a aidé Guillaume sur l'interface graphique du plateau de jeu, notamment au niveau du refresh de la grille (redessiner l'arène à chaque tour de jeu).

Guillaume Doré : Réalisation des plugins implémentant la réflexivité, Implémentation du container de jeu et de ces composants, révision du GestionnaireFrame pour adapter l'ajout des plugins via la réflexivité. Implémentation des tours de jeu et du déroulement de la partie. Adaptation des composants du moteur pour l'interopérabilité avec la nouvelle interface graphique implémentant le jeu.

Flavien Bosques : Il a tout d'abord été question de créer des plugins, notamment le plugin de déplacement aléatoire. Par la suite en janvier il a fallu mettre en place le système d'annotations qui allait permettre de marquer les plugins afin de les charger dynamiquement. La deuxième semaine de janvier a été consacré à l'écriture des autres plugins en collaboration avec mes collègues. Les derniers jours ont été le théâtre de séance de débogage intensives, où avec mon collègue Guillaume nous avons travaillé d'arrache-pied pour avoir une démo qui nous convienne le jour de la soutenance de projet. Le dernier jour, je l'ai consacré à la correction de divers bugs mineurs et à l'établissement du rapport.

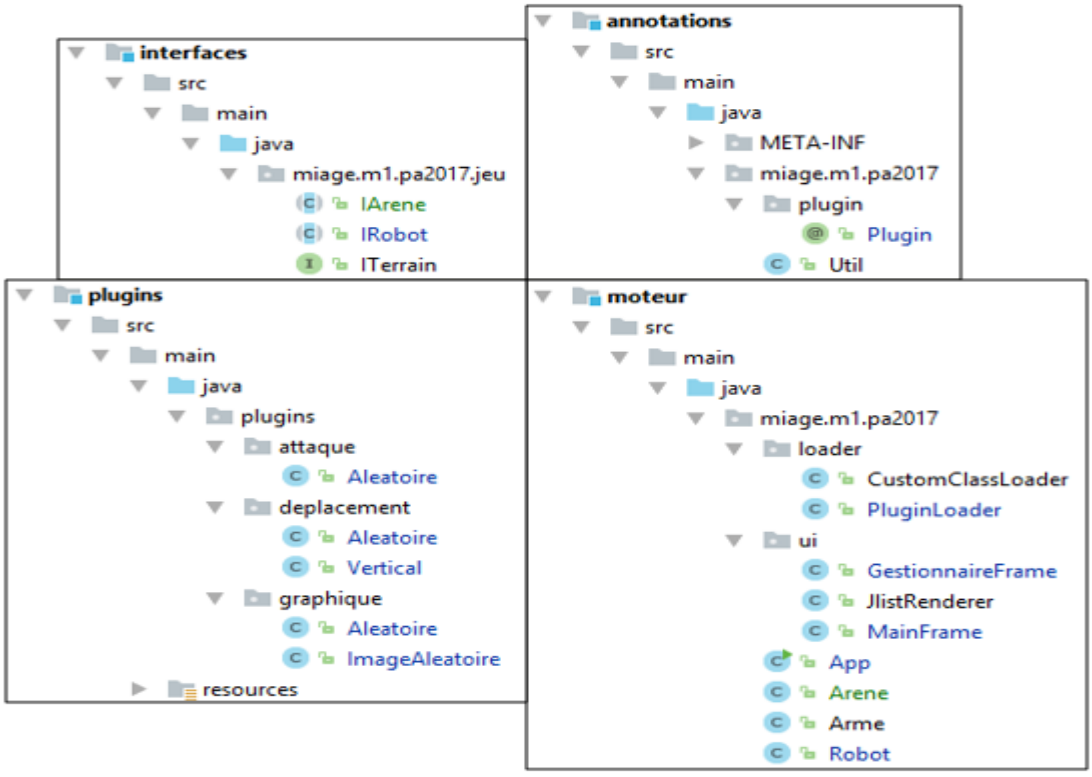
Mamadou Thiaw :
Ma participation sur ce projet commence d'abord par l'implémentation de l'architecture du jeu avec la gestion de toutes les dépendances entre les sous modules. Par la suite, j'avais crée la partie UI qui comprend deux menus, l'un pour quitter et l'autre qui donne accès à la fenêtre

“GestionnaireDePlugin”. Au niveau de l’implémentation, j’ai utilisé le patron Singleton sur le “gestionnaireDePlugin” afin d’avoir une instance unique. En parallèle, j’ai aussi utilisé le UIManager de Swing pour avoir un design flexible sur les composants graphiques (suivant le OS utilisé). De plus, j’ai choisi les boites de dialogues pour gérer les exceptions. Par la suite, j’avais implémenté notre unique annotation (@Plugin et EnumUtil) ainsi que notre premier plugin (PluginGraphiqueAleatoire qui change les couleurs). En même temps, j’avais amélioré l’UI afin de pouvoir charger les plugins via les classes du sous module Moteur que j’avais créé auparavant. En outres, j’ai complété les classes Terrain, Robot, IRobot, ITerrain, ... créées durant la mise en place de l’architecture afin de pouvoir combiner le triplé (annotations-plugin-robot).
En parallèle, je mettais régulièrement à jour le readme afin d’expliquer à mes collègues la conduite à tenir face aux modifications du projet et notamment pour la création de nouveaux plugins.

B. Architecture générale

En ce qui concerne l’architecture du projet, il existe 4 sous-modules :

- La partie interface qui est en lien avec les classes métiers.
- La partie annotations est là pour gérer le chargement de classe par annotation.
- Le moteur est formé d’une partie classe métier, d’une partie graphique (UI) et d’une partie chargement de classes (loader).
- La partie plugin est séparée en packages liés aux fonctionnalités des plugins (graphique, attaques, déplacements).



C. Les plugins

Différents plugins on était réalisés lors du projet, à savoir :

- Deux plugins graphiques,

- Plugin CouleurAleatoire ➜ il permet d'affecter à un robot carré avec une couleur (RGB) aléatoire.
- Plugin ImageAleatoire ➜ il permet d'affecter à un robot une image (.png) aléatoire parmi les images stockées dans le dossier "plugins/src/main/ressource" du projet.

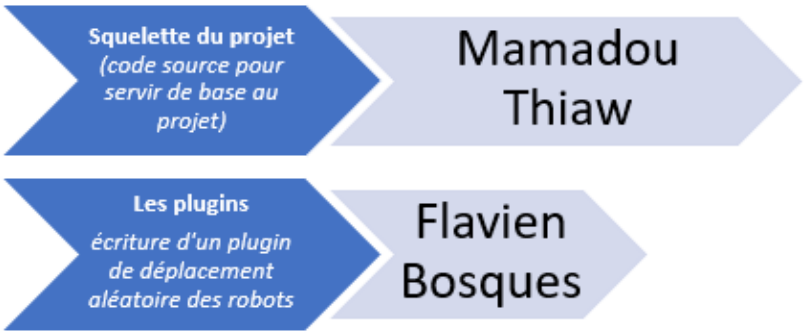
- Deux plugins de déplacement,
  - Plugin DeplacementAleatoire ➜ il permet au robot de se déplacer aléatoirement dans une direction, c'est-à-dire qu'il a une chance sur quatre d'aller vers le haut, vers le bas, vers la droite et vers la gauche.
  - Plugin DeplacementHorizontal ➜ il permet au robot de se déplacer latéralement et au hasard, c'est-à-dire qu'il a une chance sur deux de se diriger vers la droite et vers la gauche.

- Un plugin d'attaque,
  - Plugin AttaqueAleatoire ➜ il permet à un robot d'en attaquer un autre en lui infligeant des dégâts de vie lorsqu'il rentre dans son champ de vision (portée de l'attaque).

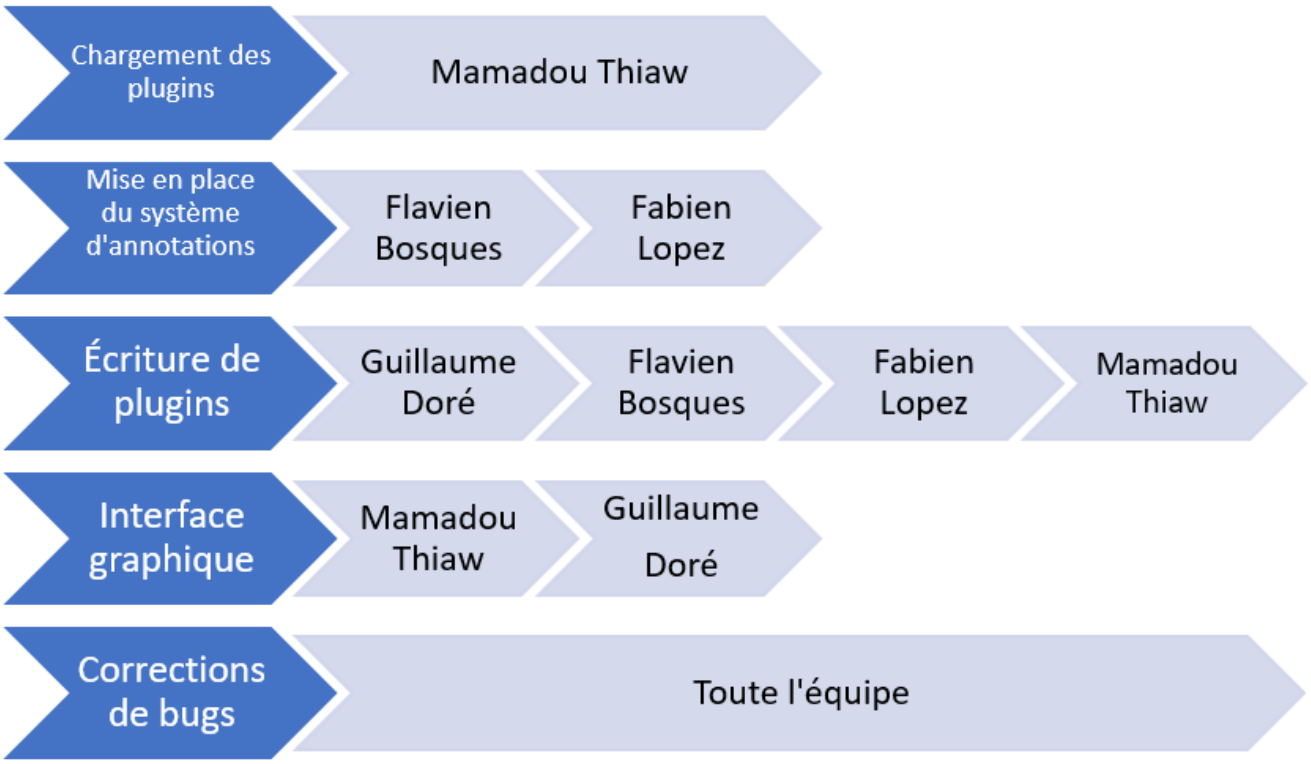
Partie III: Réalisation

A. Les grandes étapes du projet (par membre)

Le mois de décembre



Le mois de janvier



C. Les 5 aspects du projet

1.Fonctionnalité

Notre version du RobotWar permet avant le démarrage d'une partie de charger un fichier .jar de plugins avec le bouton « Gestionnaire des plugins » du menu « Les plugins » et ainsi parcourir l'ordinateur depuis un JFileChooser pour aller chercher le fichier JAR en question. Une fois la liste des plugins de sélectionnée, on ferme le gestionnaire et depuis le menu « Partie » démarre le jeu avec le bouton « Lancer partie ».

2.Chargement dynamique

Notre application charge dynamiquement nos plugins dans notre interface de gestion des plugins. Le tout avec un mécanisme d'annotation pour marquer les plugins et charger les classes par réflexivité.

3.Persistance

La notion de persistance est présente dans notre projet au niveau du Gestionnaire de plugins, en effet, lorsque l'on charge les plugins dans le menu du jeu et qu'on ferme ce menu. Lorsque

l'utilisateur voudra y retourner pour changer les plugins, ceux-ci seront toujours présent, il n'y a pas besoin de recharger le même .JAR. De même que ceux activés resteront en surbrillance. En revanche, il n'y a malheureusement pas de persistance au sein même du jeu. En effet par manque de temps et dans une volonté d'œuvrer en priorité pour pouvoir présenter une démonstration fonctionnelle lors de la soutenance de projet, nous avons dû faire l'impasse dessus.

4.Modularité

L'entièreté de notre projet est axée autour de la notion de modularité. Chaque partie du projet, que ce soit le moteur, les plugins, l'interface ou encore les annotations, chacune d'elles sont dans un sous module maven, et ils possèdent tous un pom.xml.

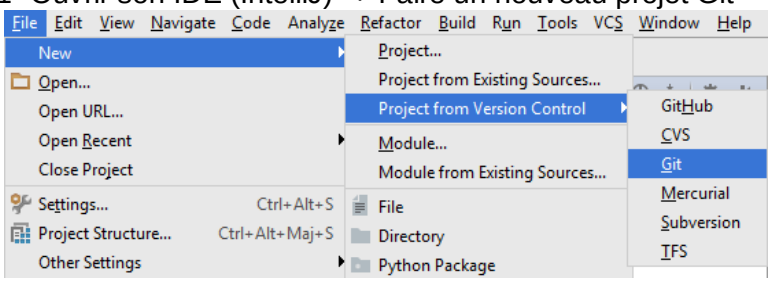
5.Suivi

Le suivi de l'avancement du projet s'est déroulé autour de réunions fréquentes dans des salons de discussion vocale sur l'outil Discord ou sur Slack pour attribuer des tâches aux membres du groupe. Dans la mesure où les échéances du projet étaient courtes, il a été décidé que l'utilisation d'outil comme Trello n'apporterait pas vraiment de plus-value.

Partie IV : Procédure à suivre pour tester le projet

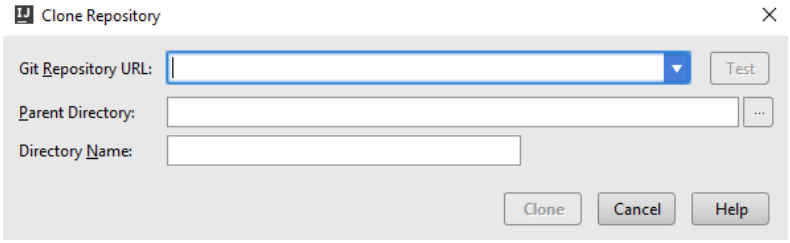
Afin de tester le projet, il faudra effectuer une suite d'action dans la console. Nous les citerons dans l'ordre dans lequel il faut les exécuter.

1- Ouvrir son IDE (IntelliJ) ➜ Faire un nouveau projet Git



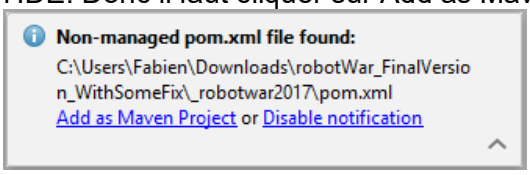
2- Cloner le dépôt Git

Une fenêtre pop-up s'affiche et il faut maintenant cloner le dépôt Git du projet à l'aide de l'URL suivant : https://fabienLopez@bitbucket.org/Flavien-MIAGE/_robotwar2017.git
Puis définir dans quel dossier placer le projet et cloner.



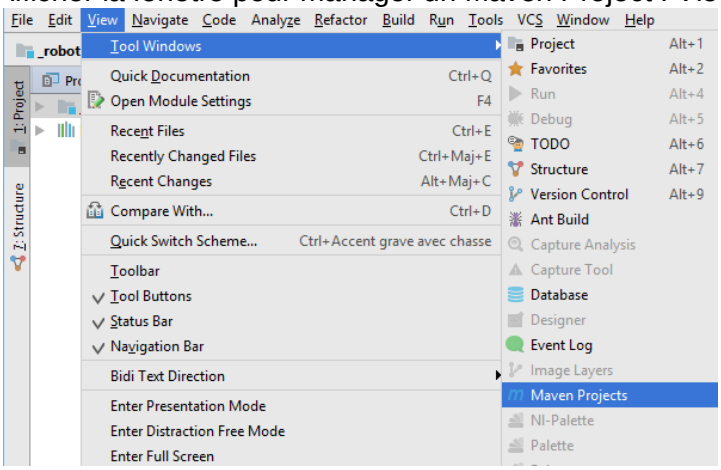
3- Faire du projet un Maven Project

Après le clonage, le projet s'ouvre et le petit encadré ci-dessous s'affiche en bas à droite de l'IDE. Donc il faut cliquer sur Add as Maven Project.

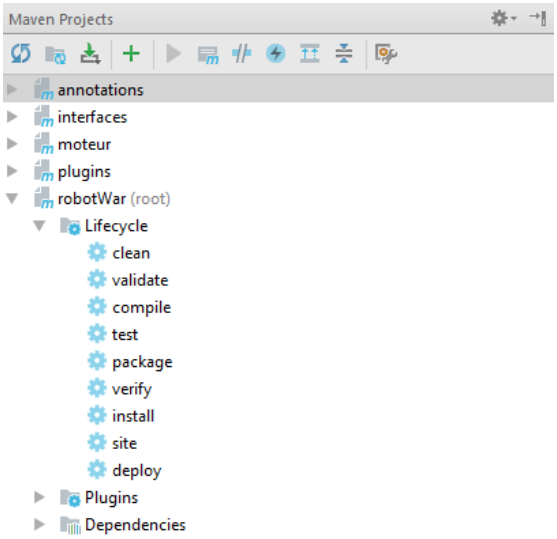


4- Compiler le projet maven

Afficher la fenêtre pour manager un Maven Project : View ➜ Tool Windows ➜ Maven Project



Puis faire un maven install :

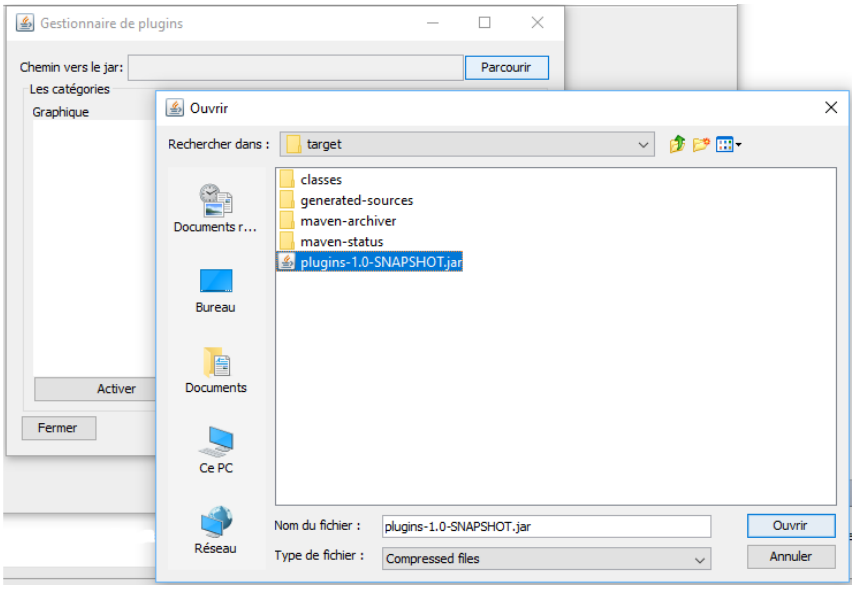


5- Lancer l'appli ➜ le main est dans la classe App.

6- Charger les plugins

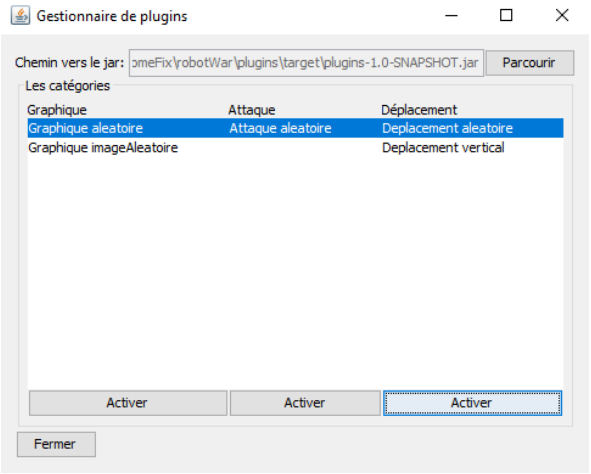
Une fois sur l'interface, dans le menu "Les plugins" choisir "Gestionnaire de plugins". Cela ouvre un pop-up pour aller chercher le JAR des plugins.

Le JAR se trouve, en partant de la racine du projet, dans plugins/target/plugins-1.0-SNAPSHOT.jar



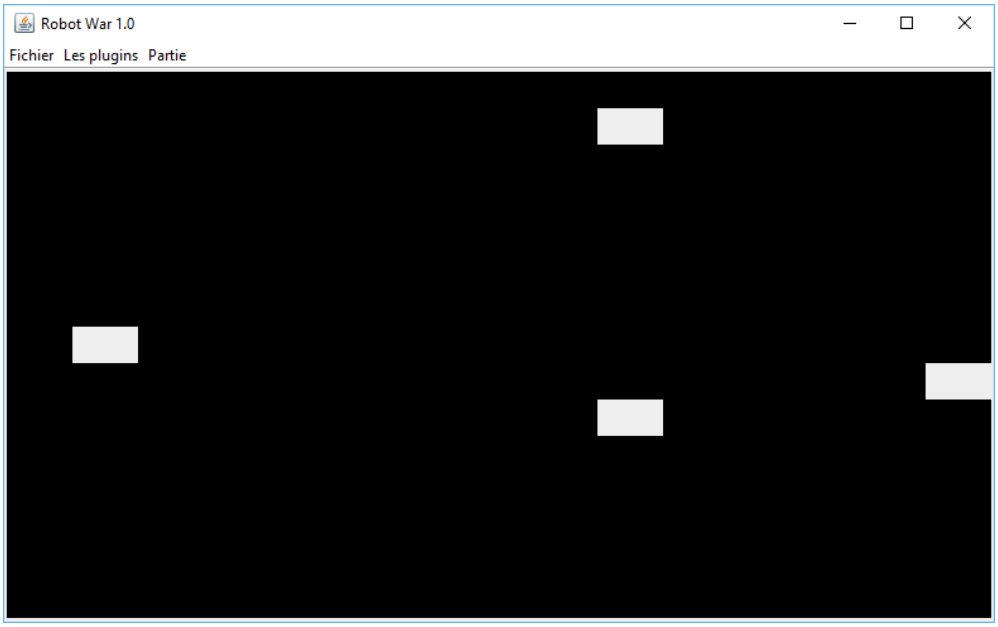
7- Activer les plugins

Seul un plugin de chaque catégorie peut être activé à la fois.



8- Lancer la Partie

Fermer le gestionnaire et démarrer le jeu depuis le menu “Partie” ➜ “Lancer partie”.



Conclusion

Ce projet nous aura permis de mettre en pratique les enseignements du cours de programmation avancée et surtout nous aura forcés à travailler dans une configuration non choisie. Le projet s'est bien déroulé. Nous avons passé un bon projet et en retirons des enseignements sur les interactions entre les membres de l'équipe. Ainsi que des notions un peu plus précises de la gestion du projet.

Vous trouverez, comme décrit dans la consigne, un guide d'ajout des plugins au même étage que ce fichier sur notre répository git.