

RAPPORT PROJET ROBOTWAR



PROGRAMATION AVANCÉE

M1 MIAGE SOPHIA ANTIPOLIS

2016 :2017

par

NICOLAS KIRCUN, TOM PHILY ET MOHAMMED TANOUTI

Dirigé par : Fabrice Huet

Table des matières

[I. Participation 3](#_Toc468919851)

[A. Nicolas Kircun 3](#_Toc468919852)

[B. Tom Phily 3](#_Toc468919853)

[C. Mohammed Tanouti 3](#_Toc468919854)

[II. Planification des étapes 4](#_Toc468919855)

[III. Procédure pour tester l’application 5](#_Toc468919856)

[IV. Les fonctionnalités de l’application 6](#_Toc468919857)

[V. Gestion des plugins 7](#_Toc468919858)

[VI. Système de sauvegarde 8](#_Toc468919859)

[VII. Dépendance antre les projets 9](#_Toc468919860)

[VIII. Gestion de projet 10](#_Toc468919861)

Ce document a pour but de décrire :

* La participation de chacun.
* La planification des grandes étapes pour chacun.
* Les procédures pour tester l’application.
* Les fonctionnalités de l’application.
* Mécanisme de gestion de plugins et de chargement dynamique.
* Le système de sauvegarde.
* Les problèmes de dépendances.
* La gestion du projet.

# Participation

## Nicolas Kircun

## Tom Phily

Mon travail au sein du groupe a été de concevoir l’architecture maven du projet, c’est-à-dire divisé le projet en trois modules. Géré les problèmes de dépendance au sein de ces modules. Permettre le lancement de l’application lors de l’exécution du fichier jar après la compilation de celui-ci.

J’ai travaillé en collaboration sur la gestion des plugins avec Mohammed Tanouti, sur la gestion de la vie et de l’énergie dans les plugins attaques et j’ai développé le plugin attaque à distance.

Le projet m’a permis d’utiliser les notions appris en cours concernant la réutilisation du ClassLoader des travaux pratiques sur le chargement de classes à partir d’un fichier jar. L’utilisation des classes internes anonymes pour lancer l’application avec un graphisme de base qui attend un décorateur

## Mohammed Tanouti

# Planification des étapes

# Procédure pour tester l’application

**Pour installer et tout lancer :**

## Sous linux :

build.sh

## Sous Mac OS :

sh build.sh

## Sous Windows :

**Si CigWin d'installé :**

sh build.sh

**Sinon :**

mvn clean

mvn package

java -jar RobotWar-jar-with-dependencies.jar

# Les fonctionnalités de l’application

## Lancement de l’application : Choix du fichier jar

Au lancement, l’application ouvre un *filechooser* dans le but de permettre à l’utilisateur à choisir le fichier jar qui contient la liste des plugins qu’il a développé pour lancer le jeu. L’application ne permet pas de sortir du *filechooser* tan que l’utilisateur n’a pas sélectionné un fichier ou kill le processus. Si le fichier sélectionner n’est pas valide et ne contient aucuns plugins, l’application va s’arrêter en envoyant une exception dans la console. Sinon, l’application se charge avec les plugins de bases et les robots s’affrontent jusqu’à qu’il n’en reste plus qu’un.

## Choix des plugins

L’utilisateur peut sélectionner les plugins au cours de la partie dans la barre des menus.

Choisir un plugin relance la partie et sauvegarde l’état du jeu dans un fichier *Sauvegarde*.*xml.* Il ne peut y avoir qu’un plugin déplacement ou attaque au cours de la partie. La sélection d’un plugin attaque ou déplacement désélectionne le précédent pour lancer la partie avec le nouveau plugin.

Cependant l’application peut cumuler les plugins graphismes.

## Sauvegarde et relance de la partie

L’utilisateur peut à tout moment appuyer sur le bouton *sauve* dans la barre de menu pour sauvegarder l’état des plugins au cours de la partie.

Si ensuite l’utilisateur appui sur le bouton *load*, la partie se relance avec les plugins sauvegardés.

# Gestion des plugins

Les plugins sont affichés dynamiquement dans la barre des menus en fonction du fichier jar sélectionné. Les classes RepositoryLoader et PluginsLoader, développées en cours, vont permettre de parcourir le jar et de charger toutes les classes qu’il contient. RepositoryLoader va ensuite classer ces classes dans des listes attaques, déplacements et graphismes en fonction des interfaces qu’elles implémentent.

Si une de ces listes est vide, l’application s’arrête et renvoie l’exception avec le message d’erreur suivant : *« aucun plugin charge »*.

L’application Swing va ensuite parcourir les listes pour ajouter dans les menus qui leur correspondent de nouvelles actions avec pour nom le plugin et pour effet de sauvegarder et de relancer la partie avec le nouveau plugin sélectionné.

# Système de sauvegarde

# Dépendance entre les projets

# Gestion de projet