# Rapport de TER

12 mai 2018

# Table des matières


**I Présentation du projet** **3**

**1 Introduction** **4**
1.1 But du projet . . . . . 4
1.1.1 Cahier des charges . . . . . 4
1.2 Notion d'agent . . . . . 4
1.2.1 Système multi-agents . . . . . 4
1.2.2 Representation dans notre projet . . . . . 4

**2 WarBot : Le mode par défaut** **5**
2.1 Principe . . . . . 5

**II Réalisation du projet** **6**

**3 Partie "Moteur"** **7**
3.1 Phase de conception . . . . . 7
3.2 De l'étude de l'ancien projet à la refonte totale du moteur. . . . . 7
3.2.1 État de l'art de l'ancien projet. . . . . 7
3.2.2 Refonte du noyau. . . . . 7
3.3 Réalisation . . . . . 7
3.3.1 Sur les unités. . . . . 7
3.3.2 La liasion avec le comportement. . . . . 8
3.4 Fonctionnalités . . . . . 8
3.5 Amélioration possible . . . . . 8

**4 Partie "Graphisme"** **9**

**5 Partie "Interpreteur"** **10**
5.1 Présentation et Attente . . . . . 10
5.2 Etude de l'ancien projet et récupération de ce qui est utile . . . . 10
5.3 Conception et implémentation . . . . . 12

**6 Partie "Game Design"** **15**

**III L'avenir du projet** **16**

**7 Amélioration possible** **17**

# Première partie

# Présentation du projet

# Chapitre 1

# Introduction

## 1.1 But du projet

L'objectif de ce projet est la réalisation d'un jeu basé sur un modèle multi-agent. L'idée générale du projet est dans la continuité du projet de l'année dernière sur le même thème. L'outil utilisé est Unity 3D, un moteur de jeu utilisé dans un grand nombre de réalisations de hautes qualités. Notre projet est utilisable sur Windows et Mac et pourrais être porté sur Android. Ce projet consiste de réaliser un jeu que l'on peut qualifier de programmeur et de permettre, notamment, à de jeunes personnes de se familiariser avec le monde de la programmation. L'utilisateur pourra donc créer un comportement pour des robots appelé "unité" afin de remplir des objectifs du jeu.

Le projet Metabot est un projet modeste réalisé à partir du logiciel Unity 3D par un groupe d'étudiant débutant dans l'utilisation de cet outil. Malgré le peu d'expérience dans la création pure de ce genre de projet, le projet actuel est le fruit d'un travail important et d'une implication entière de toute l'équipe. Le projet a donc pour unique prétention de communiquer notre amour du jeu vidéo et de la programmation.

### 1.1.1 Cahier des charges

## 1.2 Notion d'agent

### 1.2.1 Système multi-agents

### 1.2.2 Representation dans notre projet

# Chapitre 2

# WarBot : Le mode par défaut

## 2.1 Principe

Dans MetaBot, deux à quatre équipes se battent sur un terrain pour les ressources afin de survivre et d'éliminer les autres équipes afin d'être la dernière ne vie. Des ressources apparaissent sur la carte et peuvent etre converti en unité ou en soin.

# Deuxième partie

# Réalisation du projet

# Chapitre 3

# Partie "Moteur"

## 3.1 Phase de conception

Pour réaliser ce projet, nous avons réalisé une conception basé sur notre savoir en matiere de programation orientée agent et tout en gardant à l'esprit la notion de généricité qui est au coeur de notre projet. Le but de notre moteur de jeu est de permettre à des developpeur de pouvoir réaliser un mode de jeu orienté agent, de facon la plus simple possible.

## 3.2 De l'étude de l'ancien projet à la refonte totale du moteur.

### 3.2.1 État de l'art de l'ancien projet.

### 3.2.2 Refonte du noyau.

## 3.3 Réalisation

### 3.3.1 Sur les unités.

**Le corps**

**L'esprit**

Afin de pouvoir fonctionner, les unités possede 3 parties distinctes : La gestion des actions possibles, la gestion des perceptions et l'identité propre de l'unité.

**Actions.** Blablabla

**Perceptions.** Blablabla

**Identité.** Blablabla

### 3.3.2 La liasion avec le comportement.

## 3.4 Fonctionnalités

## 3.5 Amélioration possible

# Chapitre 4

# Partie "Graphisme"

# Chapitre 5

# Partie "Interpreteur"

repasser sur le texte pour ajouter exemples et illustrations !

## 5.1 Présentation et Attente

La partie "Interpréteur" est la partie la moins visible du projet MetaBot, mais il s'agit de la partie du projet servant de clé de voute du jeu. Comme dit plus tôt, la particularité de MetaBot est que le joueur, qui pourra être considéré comme le programmeur, va préparer en amont une cohésion d'équipe à travers le comportement et va pouvoir lancer un match contre une autre équipe, afin d'évaluer quel comportement sera le meilleur. L'interpreteur permet de faire la liaison entre l'éditeur du comportement ou l'utilisateur va développer son comportement, en utilisant un ensemble de d'instructions que nous avons prédéfinies et la partie moteur, ou le fonctionnement des unités est inscrit, ainsi que les différents modes de jeux.

Le langage et l'ensemble des instructions nécessaires est alimenté par l'équipe Game Design qui nous a donné des exemples de messages ou de spécificités du langage qui pourrait être nécessaire. On pouvait ensuite tous en discuter en pesant le pour et le contre, afin de définir si la fonctionnalité allait être mise en place , et de quelle façon.

## 5.2 Etude de l'ancien projet et récupération de ce qui est utile

Au départ, il a été nécessaire de remettre en place un outil permettant de récupérer un comportement, qui était uniquement graphique, dans l'éditeur afin de pouvoir le renvoyer à la partie Moteur, pour que l'ensemble des unités puissent l'exécuter. Nous avons pris connaissance de ce que l'ancien groupe avait mis en place et avons trié ce qui nous semblait correspondre à notre version du projet. Il

était obligatoire d'écrire le comportement récupéré de l'éditeur dans un fichier, afin de le récupérer , pouvoir le modifier , le déplacer , et le conserver pour plusieurs parties. La solution mise en place par l'ancien groupe pour le stockage, qui était d'utiliser un fichier XML correspondait parfaitement à notre besoin, car la syntaxe et l'organisation en nœuds de ce genre de fichiers, permettait une récupération simple et claire des instructions. Nous avons ainsi pu récupérer une bonne partie de leur système d'écriture et de lecture de leur projet, tout en adaptant l'autre partie à nos besoins.

La partie organisant les instructions à été complètement supprimée, et nous avons repensé un système plus générique et plus compréhensible pour les groupes qui vont récupérer ce projet plus tard.

## 5.3 Conception et implémentation

La construction des Instructions des unités dans la partie Interpréteur a grandement été facilitée et la liaison des instructions avec leur exécution a été décalée dans le moteur. Ainsi on ne manipule que des chaînes de caractères lors de l'accès dans le fichier, et le moteur effectue la liaison avec une structure de "Delegate" comme expliqué plus haut, rajoutant beaucoup de généricité au projet. D'un point de vue de la sécurité, celle-ci a été décalée dans l'éditeur du comportement, afin de ne pas traiter des comportements erronés , mais prévenir a l'avance et même empêcher les erreurs d'apparaitre.

Le but de l'interpréteur était de traiter des fichiers de ce type :

```
<behavior>
  <teamName>Default Team</teamName>
  <unit name="Explorer">
    <instruction>
    </instruction>
  </unit>
  <unit name="Base">
  </unit>
  <unit name="Light">
  </unit>
  <unit name="Heavy">
  </unit>
</behavior>
```

Ainsi nous avons 1 fichier par équipe écrite par l'utilisateur, chacun détaillant le comportement de chaque unité sous la même forme :

```
<behavior>
  <teamName>Default Team</teamName>
  <unit name="Explorer">
 <instruction>
      <parameters>
        <PERCEPT_ENEMY />
      </parameters>
      <message>
        <ACTN_MESSAGE_HELP>
          <Light />
        </ACTN_MESSAGE_HELP>
      </message>
      <actions>
        <ACTION_MOVE />
      </actions>
    </instruction>
    <instruction>
      <parameters>
        <PERCEPT_BLOCKED />
      </parameters>
      <actions>
        <ACTION_MOVE_UNTIL_UNBLOCKED />
```

```
      </actions>
    </instruction>
  </unit>
</behavior>
```

Le comportement d'une unité est une suite d'instructions, dont l'ordre est très important. La hiérarchie va déterminer la priorité qu'aura l'action sur le tick. Ainsi l'action la plus prioritaire dont les conditions sont acceptées sera l'action effectuée sur ce tick. Chaque instruction est toujours organisée de la même façon : - Une liste de conditions à remplir pour que l'instruction soit considéré comme acceptée - Une liste d'actions qui ne terminent pas l'action, appelées Actions non terminales, mais qui peuvent être les messages partagés entre les unités par exemple. Ces actions non terminales seront exécutés avant : - L'action terminale, qui va être exécutée à la fin du tick pour l'ensemble des unités et va être l'action qui va être "physiquement" fait par l'unité, par exemple : Avancer, Tirer, Récupérer ou donner une ressource,Se soigner , ...

Comme dit précédemment, l'interpréteur relie les 2 parties majeures du projet. Ainsi, une fonction permet de transformer un fichier ".wbt" (format des équipes WarBot) vers un comportement d'équipe utilisable par l'équipe moteur, et une fonction permettant la transformation d'un comportement récupéré de l'éditeur graphique vers un fichier XML, pour sa sauvegarde.

Un nouveau travail d'interprétation a vite été ressenti dans le projet. En effet, le jeu étant destiné à aider les jeunes à appréhender la programmation, une traduction française était nécessaire. Toujours dans l'idée de rester générique, un traducteur a été mis en place et attaché à l'ensemble des textes qui seront affichés à l'utilisateur.

//pas clair a changer La traduction fonctionne ainsi : Un premier script s'occupe de vérifier la langue qui doit être utilisée dans le fichier de configuration du jeu et va charger en mémoire un fichier de traductions :

Le fichier va être parcouru, et un ensemble de couples clé,valeurs vont être récupérés , la clé correspondant à la valeur initiale d'un bouton/ texte affiché, la valeur étant la traduction qu'il faudra affiché à la place de la clé.

Ce script est ainsi rattaché au "GameManager" , un objet Unity qui n'est pas détruit lorsque l'on change de scène. Ainsi les traductions sont conservées lorsqu'on passe de l'éditeur au menu principal , du menu principal aux cartes de jeu, ...

Un second script est lui attaché à tous les Objets du jeu qui vont devoir être traduits. Celui-ci va , à la création de l'objet auquel il est rattaché, chercher la langue actuellement affichée, récupérer comme clé le texte original de l'objet auquel le script est affilié, et va stocker la traduction, et l'afficher à l utilisateur.

En Runtime, si la langue est changée dans les paramètres du jeu, les script de traduction des objets vont mettre à jour leur langue, et récupérer la nouvelle traduction dans le "GameManager".

# Chapitre 6

# Partie "Game Design"

# Troisième partie

# L'avenir du projet

# Chapitre 7

# Amélioration possible