**17/12/2019**

Perdaens Martin

Michotte Martin

Valentin Morgan

BattleGround

Table of Contents

[Cahier des charges 2](#_Toc27037841)

[Diagramme UML 2](#_Toc27037842)

[Choix d’implémentation 2](#_Toc27037843)

[Difficultés rencontrées 2](#_Toc27037844)

[1. Implémentation de la communication Client-Serveur avec le jeu 2](#_Toc27037845)

[2. Implémentation de l’architecture MVC 2](#_Toc27037846)

[3. ? 2](#_Toc27037847)

[Pistes d’amélioration 2](#_Toc27037848)

[ Revoir complètement la structure de notre architecture MVC 2](#_Toc27037849)

[ Ajouter le bonus “radar-discovery” 2](#_Toc27037850)

[ Améliorations du système de positionnement des unités 2](#_Toc27037851)

[ Amélioration de la gestion des tirs 2](#_Toc27037852)

[Conclusion individuelle 2](#_Toc27037853)

[Martin Perdaens 2](#_Toc27037854)

[Morgan Valentin 2](#_Toc27037855)

[Martin Michotte 2](#_Toc27037856)

# Cahier des charges

# Diagramme UML du modèle

# Choix d’implémentation

## Difficultés rencontrées

## Maitrise de l’outil « Git ».

* C’était pas vraiment à proprement dit une difficulté, mais juste beaucoup de temps consacrés à une petite chose qu’un connaisseur (professeur) aurait pu nous expliquer en quelques minutes comme le traitement des branches, les branches distantes (remote branch) et comment fonctionne un merge.

## Implémentation de la communication Client-Serveur avec le jeu

* Dès que le diagramme UML était fini, nous nous sommes penchés sur la problématique de la communication entre 2 pc. On a vu plusieurs manières de l’implémenter sur internet mais le point commun avec toutes ces manières, c’est que le modèle repose sur le fait que le serveur est en quelque sorte « passif » : celui-ci attend toujours les requêtes/demandes d’un client. Et une fois qu’il reçoit une demande d’un client il lui répond.
* Pour notre jeu qui comprends 2 joueurs (donc 2 clients), il fallait trouver un moyen que lorsqu’un des 2 clients envoie une requête au serveur, celui-ci envoie une information aux **2 clients et pas juste à celui qui a fait la requête**.

## Implémentation de l’architecture MVC

* Nous ne sommes absolument pas contre MVC, au contraire c’est une très bonne idée /pratique. Le problème c’est qu’une fois que nous avons fini notre code pour la ligne de commande fonctionnel, une semaine avant la démonstration on reçoit les explications de ce qu’est MVC, l’utilité de celui-ci et comment l’implémenter (dans un code relativement simple).
* Notre grande difficulté, réside dans le fait de devoir revoir la structure du code pour implémenter MVC dans le peu de temps qui nous restaient.

## Pistes d’amélioration

## Revoir complètement la structure de notre architecture MVC

* On devrait en quelque sorte recommencer notre code à zéro en implémentant l’architecture MVC pour avoir un « vrai » MVC.
* (Martin tu sais mieux quoi dire ICI)

## Ajouter le bonus “radar-discovery”

* Une des armes (ou bonus) que nous n’avons pas réussis à implémenter c’est le bonus « radar-discovery » qui permettrai à un joueur de connaitre l’emplacement des unités ennemies (en sélectionnant une petite zone).
* Elle figure bien dans le code :
  + Dans le **gui** elle est grisée => bouton désactivée mais présent.
  + Dans la **ligne de commande** => Commentée mais présent.

## Améliorations du système de positionnement des unités

* ??

## Amélioration de la gestion des tirs

* Tir partiellement en dehors de la grille :
  + Lorsqu’un joueur tire en dehors de la grille (sauf le tire de base), par exemple le « *big-shoot* » où le joueur choisi une case et le tire touche la cache et toutes les cases adjacentes. Si le joueur choisi de tirer dans la case supérieure gauche (A1), il touchera 4 cases au lieu de 9 cases.
* Tir sur une case qui a déjà été tirée dessus :
  + On devrait peut-être implémenter un petit message disant au joueur que la case a déjà été visée afin que celui-ci ne perde pas un tour.

# Conclusion individuelle

## Martin Perdaens

* En ayant jamais programmer en Java, j’avais des crainte sur la manière de coder le projet. Au début je voyais pas comment faire mais par la suite avec les cours et le faites de chercher sur internet cela m’a permis de mieux comprendre.
* L’idée de faire un projet en groupe était chouette pour approfondir notre compréhension du langage mais j’aurais aimé avoir toute la matière avant de commencer le projet.

## Morgan Valentin

* Etant débutant en java, j’appréhendais un peu le langage. Cependant, ce projet m’a vraiment poussé à rentrer dans le langage et chaque difficulté que j’ai rencontré m’a permis d’aller plus loin dans la matière.
* Ce projet était une très bonne idée malgré que j’aurais aimé avoir aucun cours pratique et juste le projet **ou bien** un quadrimestre où l’on apprend le langage (comme pour le cours de programmation de 1ère année) et le 2ème quadrimestre où l’on a un projet.

## Martin Michotte