

DÉPARTEMENT D’INFORMATIQUE

Faculté des sciences

Université de Sherbrooke

Livrable 2 – Capture du drapeau

Par

FÉLIX-ANTOINE OUELLET 09137551

Dans le cadre du cours :

Conception de systèmes temps réel

IFT729

Travail présenté à :

Patrice Roy

Sherbrooke

11 mars 2013

Sommaire

[Produit livré 1](#_Toc353538321)

[Tests 1](#_Toc353538322)

[Problèmes rencontrés 1](#_Toc353538323)

[Conclusion 1](#_Toc353538324)

## Produit livré

## Tests

Les tests implémentés sont ceux promis dans le livrable 01. Ainsi, chaque module, Commandant, Planificateur et Navigateur, est d’abord testé pour son bon fonctionnement. Par la suite, on teste chacun des modules pour son efficacité en termes de temps. Pour ce qui est du Navigateur, on va s’assurer qu’il est en mesure de découvrir un chemin d’un extrême à l’autre d’un niveau en moins de temps

## Résultats

## Problèmes rencontrés

Le principal problème rencontré lors du développement du commandant fut l’immaturité du SDK et de l’API fournis par les organisateurs de la compétition. De fait, une liste exhaustive des problèmes et des comportements bizarres qui pourraient survenir lors d’une partie se retrouve dans le fichier *bugs.txt*. Le plus critique de ces problèmes concernent le module de communication qui éprouve parfois des problèmes lors de la livraison de messages aux commandants impliqués dans une partie. En effet, il arrive que le traitement côté serveur prenne un peu trop de temps et que cela conduise à l’arrêt abrupt d’une partie. Ceci est très problématique, car il s’agit d’un problème hors de mon contrôle. Il y a quand même de la lumière au bout du tunnel dans le fait que les organisateurs de la compétition ont reconnus qu’il y avait des problèmes au niveau du module de communication. Cependant, la nouvelle version d’AI Sandbox comportant les correctifs promis ne sera disponible qu’après la fin de la session courante.

L’autre problème qui a grandement ralenti le processus de développement fut qu’une partie du débogage du se faire en mode *Release*. En effet, quand arriva le temps de tester le commandant dans un contexte réel, il fallu demander au compilateur d’optimiser le code produit sous peine de ne pas pouvoir initialiser le commandant dans l’intervalle de temps alloué. Évidemment, le compilateur n’est pas tenu de respecter l’ordre d’exécution des instructions tel que prescrite par le développeur, il est uniquement tenu respecter l’intention du programmeur. Encore pire, il a le droit d’éliminer des instructions qui sont soient superflues soient qui ne seront jamais atteintes dans l’exécution du programme. Par conséquent, le débogueur va cruellement manquer d’informations pour aider le développeur à déterminer les erreurs présentes dans son code. Le débogage se résume donc à tenter de reconstituer l’environnement du programme lorsqu’il a planté avec les informations incomplètes que possèdent le débogueur à cet instant. Morale de l’histoire, déboguer du code optimisé se situe dans les mêmes sphères de difficulté que de déboguer du code multiprogrammé.

## Conclusion

Lorsqu’on regarde l’aspect temps réel du produit livré, on peut constater qu’il respecte les contraintes qui avaient été annoncées en début de projet. Bien entendu, l’efficacité du commandant en tant que planificateur laisse encore à désirer, mais ceci est matière à une analyse pour un autre cours soit IFT702.