RAPPORT TECHNIQUE
PRÉSENTÉ À L'ÉCOLE DE TECHNOLOGIE SUPÉRIEURE
DANS LE CADRE DU COURS LOG792 PROJET DE FIN D'ÉTUDES EN GÉNIE LOGICIEL

SENTINELLE PAINTBALL

JACQUES DUBÉ
DUBJ14068408

DÉPARTEMENT DE GÉNIE LOGICIEL ET DES TI

Professeur superviseur

Alain April

MONTRÉAL, 21 DÉCEMBRE 2011
AUTOMNE 2011

REMERCIEMENTS

Je tiens remercier Julien Lauzé, pour avoir eu la gentillesse de nous fournir le code du projet initial ainsi que la documentation qui a permis de mettre en place notre environnement de développement. Sans oublier bien sûr le support que celui-ci m'a à apporter au début du projet.

Je remercie également Stéphane Fréniatte, mon chef d'équipe, qui dans les derniers balbutiements a pratiquement sauvé le projet en le dirigeant dans une nouvelle direction. Mais également pour le soutien technique qu'il a offert concernant la librairie OpenCV et FlyCapture2 ainsi que le temps qu'il a bien voulu me consacrer.

Et finalement, je tiens remercier chaleureusement Alain April, qui a su faire preuve d'une compréhension et d'une patience incroyable pendant les temps durs du projet.

SENTINELLE PAINTBALL

JACQUES DUBÉ
DUBJ14068408

RÉSUMÉ

Une tourelle de paintball est en ce moment développée par Alain April et son équipe. L'objectif de ce projet est d'agir à titre de consultant pour analyser les composantes actuelles et apporter des modifications aux logiciels afin d'améliorer les performances de la tourelle. Et ce, afin d'améliorer suffisamment cette dernière pour éventuellement créer un nouveau club étudiant et permettre à l'ÉTS d'entrer dans les compétitions organisées pour ce type d'automate.

TABLE DES MATIÈRES

Page

LISTE DES FIGURES

Page

LISTE DES ABRÉVIATIONS, SIGLES ET ACRONYMES

GUI	: Interface Utilisateur Graphique
UI	: Interface Utilisateur
MVC	: Modèle-Vue-Contrôleur
OS	: Système d'exploitation
PFE	: Projet de fin d'étude
IA	: Intelligence artificielle
SDK	: Plateforme de développement logiciel (Software development kit)
UML	: Langage de modélisation unifié
SRS	: Document de spécification des exigences logicielles
SWEBOK	: Corpus de connaissance de génie logiciel

INTRODUCTION

Une tourelle de paintball est actuellement en développement par Alain April et son équipe. Cette tourelle a pris naissance après une activité de réingénierie d'un système existant et nous croyons que ce type d'automates pourrait ajouter un type de compétition très intéressant pour les clubs étudiants de l'ÉTS (sans compter les activités-bénéfices qui pourraient en découler, vu la popularité du paintball). Cependant, avant de soumettre la candidature du projet pour former un club étudiant, il est important d'avoir un modèle de base. Ce dernier est actuellement capable de tirer des balles, mais de façon désordonnée sur tout ce qui bouge, sans précision et sans se soucier de la quantité de balle qu'elle possède. Le défi est donc d'arriver à calibrer/modifier la tourelle afin qu'elle puisse être plus précise dans ses tirs, plus intelligente dans le choix des cibles et la quantité de munitions utilisée, plus rapide… Bref, elle doit pouvoir causer un défi pour les joueurs de paintball sur le terrain. Comme ce projet ne dure que 3 mois, il est évident qu'il ne sera possible de régler ces problèmes qu'en partie. Cependant, l'analyse du code de ce projet met en évidence un très fort couplage avec une interface utilisateur très complexe et un manque cruel de commentaire rendant la résolution de ces problèmes très complexe. Du coup, la grande priorité de ce projet sera de faire une refonte du design et une activité massive de réusinage afin de rendre le code plus compréhensible pour nos successeurs. Si le temps le permet, les fonctionnalités principales pourront par la suite être mises en place. Entre autres, le contrôle manuel et semi-manuel de la tourelle ainsi que sa calibration et si le temps le permet, les premiers balbutiements de l'IA. Finalement, il faudra maintenir une bonne documentation pour assurer le transfert de connaissances à la relève.

CHAPITRE 1

EXPLICATION DU CONTEXTE

1.1 PRÉSENTATION DE LA TECHNOLOGIE UTILISÉE

Le projet Sentinelle Paintball à nécessité l'utilisation de plusieurs technologies pour parvenir à aux résultats actuels. Pour débuter, la plateforme de développement de Qt a été sélectionnée pour le projet. Qt est un SDK à source libre développée par Nokia. Il permet de coder principalement dans le langage C++ et possède l'avantage d'être multiplateforme. C'est-à-dire que les applications développées en Qt peuvent être exécutées sous Windows, Linux Mac OS X ainsi que les appareils cellulaires Nokia (Symbian, Maemo, Nokia N9…). Pour être utilisé sur une plateforme particulière, la seule spécificité est de télécharger le code source et de le compiler sur sa machine OU de télécharger directement un installateur. L'avantage de la première méthode est qu'une fois compilé, Qt prendra en considération les paramètres de la machine sur laquelle il se trouve. De plus, Qt possède des mécanismes (tel les SIGNAL/SLOT) qui aident à contrôler les événements logiciels, une excellente documentation qui peut aisément se comparer à la documentation Java d'Oracle ainsi qu'une bonne banque de macro ainsi qu'un framework de développement complet et unique qui améliore le langage C++.

Ce dernier langage est également celui qui fut choisi pour le projet, puisque la majorité du code déjà développé et le module de vision utilisent ce langage. Du coup, on évite ainsi d'avoir recours à des adaptateurs et puisque tout est sous le même langage, on évite les problèmes! Ce qui donne une meilleure performance tant du côté logiciel que du côté du travail d'équipe lors de la mise en commun des modules. De plus, le langage C++ est supporté par FlyCapture2 et OpenCV. FlyCapture2 est le SDK ainsi que les pilotes logiciels nécessaires à l'utilisation de notre caméra Tamron. Cette caméra est d'ailleurs la même que celles utilisées par tous les clubs étudiant en robotique de l'ÉTS. De son côté, OpenCV (CV

pour *Computer Vision)* est une librairie à source libre utilisée pour le développement de la vision de notre sentinelle. Plus particulièrement la conversion des informations de la caméra en un flux de données traitable par nos filtres de vision. En fait, cette librairie contient plus de 2500 algorithmes optimisés et est très populaire chez les adeptes de la vision robotiques. Sans compter le fait que la communauté est très active (la dernière version lancée en août 2011!). Du coup, nous avons accès à une librairie qui sera encore supportée pour un bon moment et qui possède une documentation très détaillée. Il existe d'ailleurs des manuels utilisés à guise de référence dans les cours magistraux.

Finalement, nous avons utilisé Umlet pour dessiner les différents diagrammes du système. Cet outil de modélisation étant peut-être moins puissant que *Enterprise Architect* ou *Borland Together* pour le dessin des diagrammes UML, mais possédant l'avantage d'être à source libre.

1.2 REVUE DU PROJET INITIAL

Initialement, le projet était censé être extrêmement simple : un programme existant pour la gestion des contrôles de la sentinelle nous était fourni et nous devions simplement y ajouter des fonctionnalités. Ce programme possédait l'architecture en couche suivante :

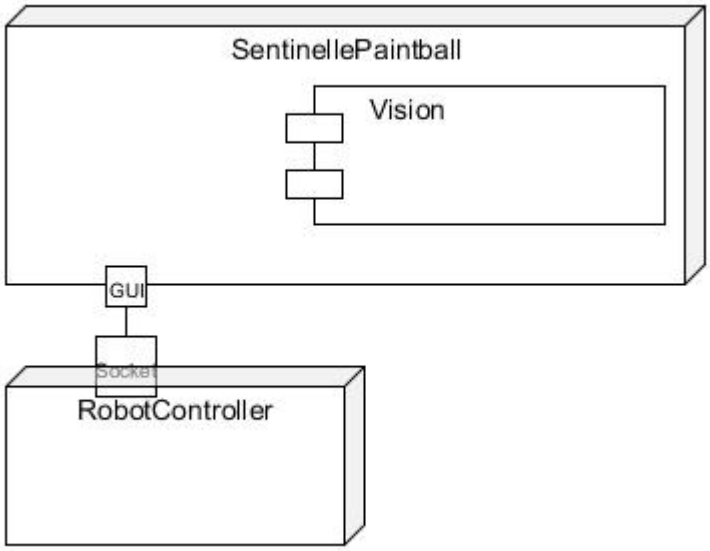

Figure 1 : Architecture de la Sentinelle

Le module qui nous était fourni, SentinellePaintball, avait comme responsabilité de récupérer les informations reçues du module de vision, les traitées et les envoyées via un socket au RobotController (gérer par une autre équipe) qui lui se chargeait de faire fonctionner le robot en conséquence. Un cas typique d'utilisation étant que le module de vision détecterait certaines cibles et les enverrait au logiciel de contrôle. Ce dernier trierait et gérerait les cibles en fonction des paramètres spécifiés par l'utilisateur et enverraient les informations nécessaires au service des servomoteurs de la sentinelle pour que celui-ci déplace le canon et fasse feu sur les cibles dans le monde réel. Pour cela, les objectifs suivants devaient être implémentés :

SPaint-OBJ-01. *Développer le système de contrôle manuel de la Sentinelle*
Permettra de valider le bon fonctionnement et la communication inter-composante de la sentinelle.

SPaint-OBJ-02. *Développer le système de calibration de la sentinelle*
Permettra d'ajuster les divers composantes de la sentinelle afin d'améliorer sa précision et vitesse de tir, de mouvement et sa gestion de munition.

SPaint-OBJ-03. *Développer les routines d'IA*
Consiste à réutiliser les modules des deux précédents objectifs afin d'automatiser le

comportement de la sentinelle afin de permettre un déploiement sur le terrain ne nécessitant pas d'intervention humaine.

Malheureusement, le programme Sentinelle Paintball ne respectait pas l'architecture initiale décrite ci-dessus (nous reviendrons sur ce point dans le chapitre 4). Le code était non commenté et très complexe et la gestion des répertoires était inexistante. Ce qui a considérablement compliqué la tâche. De ce fait, une réorientation du projet a dû être faite, telle que décrite dans la section suivante ci-dessous.

1.3 MODIFICATION EFFECTUÉE

Après de nombreuses semaines d'analyse sans aucune progression, il était évident qu'une redirection du projet s'avérait nécessaire. Après une réunion avec notre chef de projet, il n'était plus question d'ajouter des fonctionnalités au projet existant, mais de fournir un framework permettant de supporter ces fonctionnalités. Aussi, le GUI actuel avait été jugé trop lourd pour être déployé sur l'ordinateur de bord de la sentinelle et une version plus simple de l'interface devait donc être développée. Les objectifs ont donc été révisés pour devenir comme suit :

SPaint-OBJ-04. *Mettre en place l'architecture du framework de la sentinelle*

Consiste à mettre en place le patron MVC afin de briser toute dépendance entre les fonctionnalités de la sentinelle et ses UIs.

SPaint-OBJ-05. *Développer un GUI simplifié pour la sentinelle*

Consiste à développer un GUI pouvant effectuer les opérations de base de la sentinelle, mais exclure tout type d'information et de traitement de débogage utilisé par les développeurs de la tourelle.

De plus, ces objectifs devaient respecter les nouvelles exigences fonctionnelles suivantes[1] :

EF-16. *Le framework devra pouvoir supporter une interface graphique et console*

EF-17. *Le framework devra pouvoir supporter le GUI existant*

EF-18. *Le framework devra réutiliser au maximum le code existant*

EF-19. *Être portable sur plusieurs plateformes*

EF-20. *Il devrait être possible de sauvegarder les cibles et les événements du logiciel dans un fichier journal.*

Le reste de ce PFE a donc été passé à conceptualiser l'architecture de ce framework et a amorcé sont implémentation. Finalement, une fois le squelette mis en place, la conception d'un GUI simplifié a été amorcée ainsi que le développement du contrôle manuel. Finalement, des MOCs ont également été mis en place afin de simuler les comportements du module de contrôle des servomoteurs et des accès disques. Le chapitre suivant explique plus en détail l'architecture déployée.

[1] Ces exigences s'ajoutent à celles déjà établies dans le document « Liste des exigences fonctionnelles ».

CHAPITRE 2

ARCHITECTURE DÉVELOPPÉE

Ce chapitre sera principalement divisé en deux sections. La première permettra de montrer une vue globale du système développé alors que la deuxième section servira principalement à décrire les particularités et les patrons de conceptions utilisés lors de la phase de conception. Dans ce chapitre, il est supposé que le lecteur possède une certaine connaissance des concepts de programmation orientée objet.

2.1 VUE GLOBALE DU SYSTÈME

La première chose qui venait à l'esprit après l'étude du système initial était le manque de gestion de la classification des différents fichiers sources et ressources dans les répertoires du projet. Par ailleurs, les répertoires du nouveau framework ont donc été agencés de la façon suivante :

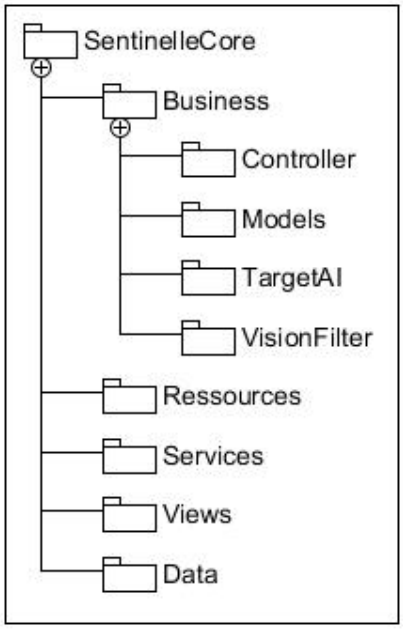

Figure 2 : Structure des répertoires

Vous pouvez déjà entrevoir dans la figure 2 les stratégies et l'architecture utilisées : une architecture 4-tiers utilisant une couche pour les classes reliées aux interfaces utilisateurs. Une pour la logique d'affaire de la solution (« Business ») qui contient tous les mécanismes importants. Une pour les services, qui couvrent principalement l'appel du module de contrôle des servomoteurs du robot et l'écoute des sockets pour la prise de contrôle distante. Et finalement, une couche « Data » pour les accès disque afin de pouvoir stocker les différents *scripts* nécessaires. Bien entendu, la couche qui est particulièrement intéressante est la couche « Business ». On peut constater qu'un nouveau module s'y est ajouté : l'intelligence artificielle (« TargetAI »). Pour le moment, ce paquet ne contient pratiquement rien, mais il a été placé dans le but d'y insérer les pipelines et les algorithmes de gestion de contrôle des cibles lorsque le moment sera venu. Les trois autres paquets contiennent les couches nécessaires à l'implémentation du patron MVC (« Models » et « Controller », la section suivante détaillera davantage les propriétés de ces deux paquets) et le module de contrôle de la vision, qui contient le code développé dans le programme originel.

Finalement, vous trouverez à la page suivante un diagramme du système développé qui montre bien les relations entre les différentes composantes du nouveau système :