M1 ALMA/ATAL
X7II030

Université de Nantes
2012/2013

Projet de travaux pratiques

I-rover

Une plateforme de prototypage de comportement de rovers

Contexte

L'aspirateur Roomba™ et la tondeuse à gazon Automower™ sont deux exemples de *rovers*, c'est à dire des robots exécutant une tâche en « autonomie » dans un environnement inconnu à l'avance. Bien que relativement autonomes, ces robots — comme leurs concurrents — ne sont pas capables de se déplacer de façon réellement intelligente dans un environnement qui peut être encombré (cas de l'aspirateur dans une pièce pleine de jouets), ou occupé par des éléments dangereux ou délicats (*e.g.*, présence d'un chien ou d'un parterre de fleurs sur une pelouse, dans le cas de la tondeuse à gazon). La page `http://electronics.howstuffworks.com/gadgets/home/robotic-vacuum2.htm` décrit succinctement le fonctionnement du Roomba™ et les contraintes que cela impose. De même, la page `http://science.howstuffworks.com/environmental/green-tech/sustainable/automower-solar-hybrid1.htm` montre les limites du système de navigation de l'Automower™, pour lequel il faut, en particulier, délimiter la zone de travail à l'aide d'un fil métallique.

On souhaite réaliser un logiciel permettant de tester des stratégies d'exploration pour un robot muni de différents senseurs.

Travail demandé

Le programme Tiled [1] permet de créer des terrains à partir de petits « carreaux » aux propriétés diverses (eau, buissons, rochers, ...) et d'exporter le résultat en XML au format TMX. Le but du projet est de créer une application permettant de :

1. Charger et afficher graphiquement un terrain fourni au format TMX ;
2. Charger une stratégie d'exploration écrite en Python (On utilisera la capacité qu'offre Python d'étendre une application écrite en C/C++ avec ce langage) ;
3. Exécuter une mission définie à l'avance en utilisant la stratégie d'exploration chargée au préalable.

Le robot *rover* devra disposer d'actuateurs et de senseurs en rapport avec la mission à effectuer au choix des développeurs. La table 1 propose certains exemples de missions, senseurs et actuateurs. Elle n'est en aucun cas limitative. D'autres exemples de senseurs sont visibles sur des sites tels que Roboshop. Les étudiants sont encouragés à faire preuve d'inventivité dans la définition de nouvelles missions et senseurs/actuateurs [2].

Les bibliothèques wxWidgets, ClanLib et Box2D sont installées localement au CIE dans l'arborescence `/comptes/goualard-f/local/{bin,include,lib}`. Cependant, le choix des bibliothèques utilisées pour réaliser le travail est laissé à l'appréciation des étudiants à la condition que l'application résultant soit compilable et exécutable au CIE. Les entorses à cette règle devront être exceptionnelles et seront à négocier *à l'avance* au cas par cas avec les encadrants de travaux pratiques.

Notation

Les projets seront notés suivant plusieurs critères :

– Qualité du code écrit (modularité, commentaires pertinents, extensibilité, maintenabilité, ...) ;

1. Disponible au CIE avec le chemin d'accès `/comptes/goualard-f/local/bin/tiled`.
2. On peut même envisager des senseurs « réels » dont la précision est entâchée d'une marge d'erreur.

Missions
Explorer chaque case « accessible » du terrain (cas d'un aspirateur autonome, par exemple)
Retrouver un objet ayant certaines caractéristiques (couleur, taille, ...)
Détruire certains éléments, éventuellement mouvants et/ou agressifs
Senseurs
Module GPS indiquant la position absolue sur la carte
Boussole magnétique indiquant le Nord
Altimètre
Télémètre infra-rouge courte-portée
Caméra (infra-rouge, lumière naturelle, ...)
Télémètre ultra-sonique longue portée
Gyroscope
Odomètre (pour déterminer la distance parcourue par le robot)
Actuateurs
Arme
Bras articulé
Lame pousseuse (type bulldozer)

TABLE 1 – Quelques exemples de missions, senseurs et actuateurs

- Qualité de de la documentation du code et des fichiers Doxygen associés ;
- Qualité de la documentation utilisateur (manuel fourni) ;
- Robustesse du code assurée par des assertions et des tests unitaires (utilisation de CppUnit) ;
- Déployabilité (utilisation correcte des *autotools*) ;
- Qualité du logiciel fourni : utilisabilité de l'interface, originalité de la mission, richesse des possibilités du robot, pertinence des stratégies d'exploration, ...

Une plus grande attention sera apportée à la qualité du développement qu'à la complétion de l'application.

Organisation

Tous les projets devront être rendus sur madoc sous la forme d'une archive tarrée/gzippée[3] au plus tard le **lundi 17 décembre 2012 à 23h55**. L'archive déposée devra contenir l'intégralité du code source ainsi que le manuel au format LaTeX et PDF. Le langage de programmation de l'application sera C ou C++.

Le projet peut se décomposer en cinq grandes parties :

- Interface graphique de l'application ;
- Lecture du terrain au format `TMX` et affichage graphique ;
- Prototypage Python ;
- Définition du robot, des senseurs, des actuateurs et déplacement du robot sur le terrain en fonction de la stratégie Python ;
- Manuel.

À l'intérieur d'un même groupe de travaux pratiques, un projet sera mené à bien par une *équipe* d'étudiants qui s'identifiera comme telle auprès de l'encadrant de travaux dirigés lors de la première séance dévolue au projet. La répartition des tâches au sein d'une équipe est laissée à l'appréciation de ses membres, les enseignants se réservant le droit de vérifier à tout moment que chaque étudiant a bien une tâche attribuée et menée à bien.

Le projet de chaque équipe sera hébergé dans la Forge du CIE. L'utilisation correcte des facilités de gestion de version de la Forge fera partie de la notation.

3. Manipulation d'archives :
   - Création de l'archive `titi.tar.gz` à partir du répertoire `titi` : `tar -vzcf titi.tar.gz titi/`
   - Récupération du contenu de l'archive `titi.tar.gz` : `tar -vzxf titi.tar.gz`