Rapport de POO : Simulateur de robots pompiers

Cyril Dutrieux, Mathias Biehler, Mahmoud Bentriou

20 novembre 2014

Table des matières

1 Présentation du programme **1**

2 Choix d'implémentations **1**
2.1 Paquetages 1
2.2 Collections 1
2.3 Classes et héritage 1

3 Attentes vis à vis du produit livré **2**

1 Présentation du programme

Le but de ce TP libre est de programmer un simulateur de robots pompiers. Ces derniers doivent éteindre tous les incendies d'une carte, affichée par une interface graphique qui nous est fourni. Le simulateur doit afficher les actions des robots dans le temps, et les prises de décision à propos des actions des robots doivent être pris en temps réel. Tout cela doit être développé selon les concepts de la programmation orientée objet.

Pour executer le programmer on utilise un script bash qui laisse le choix du manager à utiliser ainsi que la carte :
./simulation.sh [numéro manager] [MAP]
où [numéro manager] est le numéro du manager à utiliser (2 ou 3) et [MAP] est le nom d'un fichier carte situé dans le dossier cartes.

2 Choix d'implémentations

2.1 Paquetages

La source est décomposée en plusieurs paquetages :
- simulation : regroupe les fichiers propre à la simulation : données, événements élémentaires et l'affichage
- managerPack : regroupe les différents types de Manager
- elements : regroupe les éléments de la carte : robots, incendie ainsi que algorithme de plus court chemin
- environnement : Regroupe dans quel environnement les éléments évoluent (Carte etc..)
- IHM : Paquetage d'affichage graphique fournie en bytecode (dans le fichier bin)

2.2 Collections

Au niveau du choix des collections, nous avons choisi d'utiliser TreeSet<> pour stocker les événements du simulateur à executer : cette collection allie à la fois facilité d'implémentation

dans notre cas (quand on ajoute un événement, il est déjà trié dans la liste) ainsi qu'un cout algorithmique acceptable (en $O(log(n))$).

L'algorithme de plus court chemin A* a besoin d'une file de priorité, nous avons donc utilisé la collection PriorityQueue<>.

2.3 Classes et héritage

On a voulu respecter au mieux les différentes dépendances entre classes selon le graphe UML du sujet. La classe DonneesSimulation possède ses éléments : la carte qui possède ses cases, les robots et les incendies.

3 Attentes vis à vis du produit livré

Le programme s'execute correctement. Il affiche les déplacements des robots, les incendies et les incendies éteints, et indique par le biais d'un point d'interrogation quand le robot est libre.

Le Manager2 est le manager naif et Manager3 est un manager moins naif qui envoie chaque robot sur l'incendie le plus proche.