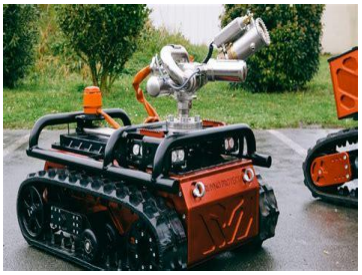
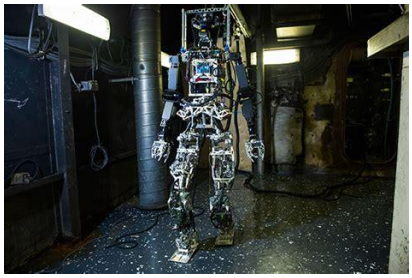
PROJET D'ALGORITHMIQUE ET PROGRAMMATION

Simulation d'une équipe de robots pompiers[1]

(à gauche : Drones (lexpress.fr) ; au centre : Colossus le robot pompier made in France qui a participé à l'extinction de l'incendie de Notre-Dame (numerama.com) ;à droite : SAFFIR le robot pompier de l'US Navy (infopompiers.com))

L'objectif de ce projet est de développer en Python un programme permettant de simuler une équipe de robots pompiers évoluant de manière autonome pour éteindre un feu.

1. Modalités

Ce projet doit être réalisé en dehors des TD, à l'exception des deux séances prévues. Pour vous éviter de prendre du retard, plusieurs jalons indicatifs sont proposés à la fin de l'énoncé.

2. Présentation du problème

L'organisation de secours lors d'une catastrophe majeure est un problème important. En particulier, les acteurs présents sur le lieu de la catastrophe peuvent être des humains, des robots, *etc*. On cherche donc actuellement à développer des moyens informatiques permettant de faciliter et de guider le travail des sauveteurs sur le terrain (voir par exemple la compétition Robocup Rescue[2]). Dans ce travail, nous allons plus modestement simuler le travail d'un groupe de robots pompiers chargés d'éteindre un feu de forêt. Les caractéristiques du problème sont détaillées dans ce qui suit.

---

[1] Ce projet est directement inspiré d'un sujet donné à l'ENSIMAG lui-même inspiré d'une idée originale de Ch. Garion (ISAE)

[2] http://www.robocuprescue.org/

3. La carte

La carte du terrain d'intervention est représentée par une **matrice** de *nl x nc* **cases** (*nl* et *nc* pouvant aller de 20 à plusieurs milliers), correspondant à une discrétisation du terrain réel.

Chaque **case** de la matrice représente une parcelle de terrain et possède un certain nombre de propriétés. Les propriétés auxquelles on s'intéresse sont les suivantes :

- *nature du terrain*, classée selon (une partie de) la nomenclature CORINE Land Cover (1994), et pouvant prendre les valeurs (exclusives) suivantes : terre arable, culture permanente, pâturage, terre agricole hétérogène, forêt, broussailles, roche nue, terre humide, lac, cours d'eau ;
- *altitude de la parcelle* (en mètres) ;

Pour simplifier le problème, on supposera que chaque case de la matrice représente une portion de terrain carrée de 100 m de côté.

Dans le projet, on dispose de 4 cartes toutes disponibles sur Arche.

4. Les robots

Les **robots** sont des « pompiers élémentaires » qui peuvent se déplacer et éteindre un feu ; ils connaissent la *carte* sur laquelle ils se déplacent et leur *position* sur cette carte.

À chaque unité de temps, un robot peut *larguer* une certaine quantité d'eau ou de gaz pour participer à l'extinction du feu ou se *déplacer* d'une case vers le feu. L'extinction totale d'un feu peut prendre un certain temps.

Nous considérons deux types de robots : **les robots volants** et **les robots terrestres**.

Dans l'application, nous considérons comme robots volants les **drones** aéroportés et comme robots terrestres les robots à **chenilles**, les robots à **roues** et les robots à **pattes**.

Les fonctionnalités auxquelles doivent répondre les robots, *a minima*, sont les suivantes :

- Tout **robot** doit pouvoir décider à un instant t, comment il doit *agir* :
    - S'il est *voisin* (1 arête en commun) de la case en feu, il effectue un *largage* sinon, il se *déplace* vers la cible en feu. Un robot ne peut effectuer qu'un déplacement ou qu'un largage par unité de temps (mais pas les deux en même temps).
        - Lors d'un largage : un robot diminue son *réservoir* d'un volume correspondant à sa *puissance* de largage. L'*intensité* du **feu** diminue

alors d'une *quantité* égale à la quantité larguée / 1000, si le fluide est de l'eau et de 0.1 si le fluide est du gaz. Quand le réservoir est vide, le robot est retiré du groupe de robots actifs..

- Lors d'un déplacement : un robot ne peut se déplacer que vers une case voisine (1 arête en commun), dans les limites de la carte et si elle est *accessible* au robot. Les robots suivent un mode de déplacement "glouton", décrit section … Plusieurs robots peuvent être presents sur une même case.

- Si un robot dépasse son temps *d'autonomie*, il est retiré du groupe de robots actifs.

4.1. Les robots volants

Spécificités des robots volants :

- Pour pouvoir faire un largage, les robots volants doivent être au-dessus du feu, et non dans une case voisine.
- L'ensemble des terrains sont accessibles aux robots volants.

Drone aéroporté : Il a une autonomie de 50 unités de temps. Son réservoir d'eau contient 400 litres au maximum et chaque largage le vide de 100 litres.

4.2. Les robots terrestres

Spécificités des robots terrestres :

- Les robots terrestres ont des capacités de déplacement qui peuvent être limités soit par la nature du terrain soit par un dénivelé trop important. À chaque type de robot de terrain, sont associés une liste de terrains impraticables et un dénivelé maximal.

Robot à chenilles : Il peut se déplacer sur des pentes ayant un dénivelé de 20 m ou -20 m maximum, sur tout type de terrain sauf les lacs et les cours d'eau. Son réservoir d'eau contient 500 litres au maximum et chaque largage le vide de 100 litres. Il a une autonomie de 50 unités de temps.

Robot à roues : il peut se déplacer sur des pentes ayant un dénivelé de 5 m ou -5 m maximum, sur tout type de terrain sauf les terres humides, les lacs et les cours d'eau. Son réservoir d'eau contient 100 litres au maximum et chaque largage le vide de 25 litres. Il a une autonomie de 150 unités de temps.

Robot à pattes : il peut se déplacer sur des pentes ayant un dénivelé de 10 m ou -10 m maximum, sur tout type de terrain sauf les lacs et les terres humides. Il utilise du gaz comme moyen d'extinction, son réservoir contient 100 litres de gaz au maximum et chaque largage le vide de 5 litres. Il a une autonomie de 100 unités de temps.

4.3. Mode de déplacement des robots

- un robot se déplace toujours d'une seule case à chaque unité de temps ;
- un robot se déplace en ↱ ou ↰ ou ↵ ou ↳ s'il le peut, c'est à dire selon une direction verticale (vers le Nord ou vers le Sud) en se rapprochant du feu, puis lorsqu'il est sur la même ligne que le feu, selon une direction horizontale (vers l'Est ou vers l'Ouest) pour se rapprocher du feu.
- si l'état du terrain ne lui permet pas de suivre cette stratégie, il emprunte la première case qui convient en tournant dans le sens des aiguilles d'une montre à partir de la case qu'il aurait dû emprunter (quitte à rallonger le trajet, trouver le chemin le plus court est un autre problème).

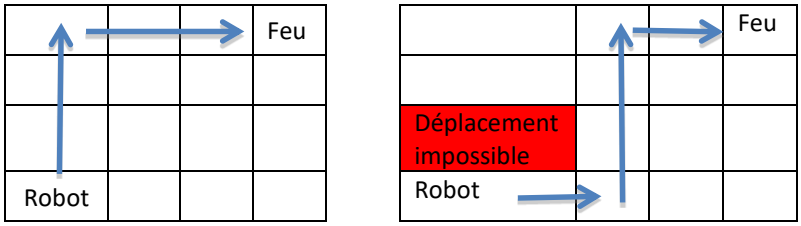


Tableau 1 : Exemple de déplacement d'un robot

5. L'incendie

On considère **l'incendie** limité à une case de *position* connue et son *intensité* diminue sous l'effet du largage mais ne peut augmenter. A l'état initial, l'incendie a une intensité par défaut de 5. Enfin, on considère que n'importe quelle case peut être incendiée.

6. La simulation

Voici le déroulement du protocole[3] appliqué pour déterminer les tâches des robots dans le cadre de notre **simulation**:

Au lancement de la simulation on fournit :

- Un fichier de données contenant la carte associée ainsi que le type et la position initiale des robots.
- La position de l'incendie (coordonnées de la case en feu)
- La *durée* maximale de la simulation en nombre d'unités de temps

Au cours de la simulation :

- Chaque robot réalise une action (largage ou déplacement, selon sa position). Si un robot est bloqué, ou s'il est à sec (réservoir vide ou autonomie insuffisante) il est retiré des robots actifs.
- Les illustrations sont mises à jour à chaque unité de temps.
- La simulation s'arrête quand:
    - Le feu est éteint : succès
    - Il n'y a plus de robots actifs ou la durée maximale est atteinte : échec

PROJET A RENDRE

Objectifs

Le programme développé en POO Python doit permettre de lancer une simulation du comportement des robots face à un incendie.

Entre autres, l'application développée devra permettre de :

- définir un environnement de simulation à partir d'un fichier décrivant une carte, la situation initiale des robots et leurs propriétés, le tout sous le format spécifié en annexe
- lancer la simulation des déplacements et des actions des robots
- visualiser la simulation à chaque étape par un affichage de l'emplacement des robots sur la carte à chaque étape (c'est à dire à chaque unité de temps)
- afficher le temps total (nombre d'unités de temps) mis pour éteindre l'incendie s'il a été éteint

[3] La stratégie des tâches affectées aux robots a été largement simplifiée par rapport à des protocoles tels que le protocole Contract Net (Smith, 1980)

Ressources fournies

- Des cartes fournies dans le format décrit en annexe
- Des exemples d'exécution du programme
- Une fonction de lecture d'une ligne dans un fichier *lire_ligne.py* permettant d'ignorer les lignes de commentaires et les lignes blanches
- Un exemple de programme principal lançant la simulation sur une des cartes fournies

Démarche

Séance de TD 1 :

La première séance de TD (en salle banalisée), consiste à conceptualiser le projet, c'est-à-dire :

- Déterminer la structure du programme : quelles classes, sous-classes, qu'elles sont leurs interaction
- Déterminer le contenu de chaque classe : attributs et méthodes. Ce contenu pourra éventuellement être mis à jour lors du développement des algorithmes (=> méthodes intermédiaires)
- Mise en place de la documentation des classes et des méthodes (entrées et sorties)

Si le temps le permet :

- Répartition des tâches pour la partie de programmation
- Réflexion sur l'algorithmique des méthodes les plus complexes

Séance de TD 2 :

La seconde séance de TD (en salle info) est consacrée à la programmation du projet. Il est conseillé de prioriser les algorithmes plus « complexes ».

Rendu final :

Le respect des bonnes pratiques de programmations telles que décrites au cours de l'année (dont dans le cours d'introduction au S5) est essentiel.

Références

- RoboCup Rescue. http://www.robocuprescue.org/.
- Commission of the European Communities : Corine land cover. http://www.eea.europa.eu/publications/ COR0-landcover/at_download/file, 1994.
- R.G. Smith : The contract net protocol : High-level communication and control in a distributed problem solver. *Computers, IEEE Transactions on*, C-29(12) :1104 –1113, dec. 1980.

Annexe

Format de description des conditions initiales

Un exemple de format de description des données dans un fichier texte est proposé Figure 1.

```
# Un exemple de carte 8x8

8 8
# l0
TERRE_ARABLE 100
TERRE_ARABLE 100
TERRE_ARABLE 100
TERRE_ARABLE 100
TERRE_ARABLE 100
TERRE_ARABLE 150
TERRE_ARABLE 200
TERRE_ARABLE 300
# l1
TERRE_ARABLE 100
TERRE_ARABLE 100
TERRE_ARABLE 100
TERRE_ARABLE 100
ROCHE_NUE 100
TERRE_ARABLE 150
TERRE_ARABLE 200
TERRE_ARABLE 300
# l2
TERRE_ARABLE 100
TERRE_ARABLE 100
LAC 100
LAC 100
TERRE_ARABLE 100
TERRE_ARABLE 150
TERRE_ARABLE 200
TERRE_ARABLE 300
...
# l3
TERRE_ARABLE 100
TERRE_ARABLE 100
LAC 100
LAC 100
TERRE_ARABLE 100
FORET 150
FORET 200
TERRE_ARABLE 300
# l4
TERRE_ARABLE 100
TERRE_ARABLE 100
LAC 100
TERRE_ARABLE 100
TERRE_ARABLE 100
FORET 150
FORET 200
TERRE_ARABLE 300
# l5
TERRE_ARABLE 100
TERRE_ARABLE 100
COURS_D_EAU 100
TERRE_ARABLE 100
TERRE_ARABLE 100
TERRE_ARABLE 150
TERRE_ARABLE 200
TERRE_ARABLE 300
# l6
TERRE_ARABLE 100
...
TERRE_ARABLE 100
COURS_D_EAU 100
COURS_D_EAU 100
TERRE_ARABLE 100
TERRE_ARABLE 150
TERRE_ARABLE 200
TERRE_ARABLE 300
# l7
TERRE_ARABLE 101
TERRE_ARABLE 102
TERRE_ARABLE 103
COURS_D_EAU 100
TERRE_ARABLE 104
TERRE_ARABLE 150
TERRE_ARABLE 200
TERRE_ARABLE 500

# Robots
2
3 3 DRONE
4 7 CHENILLES
```

Figure 1 : Format des données du simulateur. Ici un exemple de carte 8 × 8 avec 2 robots. Tout ce qui suit un caractère # est un commentaire, et les lignes blanches sont ignorées.