


	<p style="text-align: center;">Projet Informatique C</p> <p style="text-align: center;">ISEN CIR1</p>	

Déplacement d'un robot en 2D

Ce projet a pour objectif la réalisation en langage C d'une simulation de déplacement (en 2D) d'un robot dans un environnement inconnu.

L'environnement est un appartement comportant plusieurs pièces. Le robot connaît sa position, mais le plan de l'appartement lui est inconnu. Le but est de trouver la porte de sortie (il en existe une seule et il ne la détecte que quand il la « touche »).

Le comportement du robot sera extrêmement simple: il est muni uniquement d'un capteur tactile à l'avant, c'est-à-dire qu'il ne détecte un obstacle que lorsqu'il le rencontre au cours d'un déplacement.

Cahier des charges :

- Le plan de l'appartement est «maillé», c'est-à-dire qu'il se compose de «cases» qui sont soit libres, soit occupées (par un mur x ou par la porte S). Le cahier des charges ne fixe pas la taille du plan : il est fixé à la lecture du fichier qui le contient (dans l'exemple donné en pièce jointe, il fait 50x40 mailles). Le plan (qui peut changer d'une exécution à l'autre) est lu à partir d'un fichier texte extérieur au programme (cf. ex. en pièce jointe).

Il existe une porte d'entrée (et une seule) qui occupe une maille (et une seule). Elle est obligatoirement bordée (sur 2 côtés) par un mur, mais pas forcément le mur extérieur.

L'appartement est «fermé» par des murs extérieurs (contenant éventuellement la porte d'entrée), c'est-à-dire que le robot ne peut pas sortir du plan.

Le nombre de mailles en hauteur x largeur est indiqué sur la première ligne du fichier, par ex. 50:40 signifie que le plan qui suit fait 50 mailles en largeur x 40 en hauteur.

- Le robot occupe une maille et une seule : il connaît à tout instant sa position (x,y) dans le « plan ». Il est toujours orienté dans une des 4 directions suivantes : N, E, S, W. Il ne connaît pas la position (x,y) de la porte ni la taille de l'appartement (c'est-à-dire qu'il ne sait pas si le mur qu'il longe est extérieur ou pas). La position de départ (lettre D) du robot est fixée arbitrairement sur le plan initial.
- Les déplacements du robot sont «discrets», c'est-à-dire que le pas de déplacement est d'une maille (distance=1). Pour ce déplacement élémentaire, il peut se déplacer dans les 4 mailles voisines, c'est-à-dire selon une des 4 directions possibles (pas de déplacement en diagonale).

Chaque déplacement élémentaire du robot doit incrémenter un compteur. C'est la valeur de ce compteur à la fin (robot à la porte de sortie) qui permet de mesurer l'efficacité des algorithmes que vous avez mis en œuvre.

- Le robot « sait » qu'il rencontre un obstacle quand la maille suivante, dans sa

direction de déplacement, est occupée (ce qu'il saura en interrogeant le « plan »). Ceci simule un capteur tactile à l'avant du robot. **NB** : ce qui précède implique que le robot ne peut pas détecter ce qu'il a sur les côtés ou derrière lui.

- Le robot peut détecter la porte de sortie si elle est dans une des 4 mailles voisines. Ceci simule un capteur spécifique multi-directionnel, qui reconnaît la porte quand elle est juste à proximité.
- Toute amélioration de l'«intelligence» du robot est la bienvenue (par exemple, mémorisation de la position des mailles visitées, détermination «intelligente» de la position des pièces de l'appartement et des «portes» entre elles), mais ceci n'est pas obligatoire. Le but est de minimiser les déplacements.
- Une interface graphique est demandée (cf. remarques en fin de ce texte). Le code correspondant à cette interface doit absolument être séparé du code concernant le comportement des robots : on doit pouvoir changer de type d'interface graphique, sans intervenir dans le reste du code. Dans cette interface, en plus de la visualisation des éléments du plan et du déplacement du robot, il est demandé d'indiquer le compteur de déplacements. Il est nécessaire également de pouvoir arrêter l'application à tout moment par un clic sur l'icône d'arrêt (croix) habituelle.

Des améliorations de l'interface sont possibles : touche «Pause», musique ou sons par exemple, ... mais pas obligatoires.

Contraintes de programmation :

- Le programme devra être compilé (par *make*) et s'exécuter sous Linux.
- Le programme devra bien sûr être conçu selon une décomposition « procédurale » (en fonctions) et les règles de bonne programmation (pas de variables globales, nom des variables explicite,...) devront être respectées.
- Votre code doit être commenté et lisible (indenté, aéré,...).
- Pour juger votre programme, le correcteur fournira un fichier «*appart.txt*» contenant le plan de l'appartement, sous forme de chaîne de caractères, avec : les murs notés par un caractère "x" minuscule, le point de départ du robot par un "D" majuscule. Ex. fichier joint.

Livraisons et tests demandés :

- Ce projet est noté individuellement. Les échanges d'informations (verbales) sont autorisés, mais tout échange ou recopie de code est interdit.
- Un rapport de fin de projet devra décrire le fonctionnement global du programme, donner quelques informations sur les techniques mises en œuvre ainsi que sur résolution des problèmes techniques les plus difficiles rencontrés en cours de projet.
- Les sources et fichiers nécessaires à la compilation et l'installation devront être fournis.
- Une démonstration du programme aura lieu en fin de projet. Le programme sera jugé sur le nombre de déplacements nécessaires pour que le robot trouve la sortie sur trois plans d'appartement non connus à l'avance : un plan simple (porte de sortie sur mur extérieur, pas de murs en « îlots »), un moyen (porte de sortie sur mur extérieur, « îlots »), un difficile (porte de sortie sur un « îlot »).

Déroulement : projet de 30h encadrées (6 séances de 3h, 3 séances de 4h)

- Questions générales sur le sujet lors de la 1^{ère} séance
- Présentation d'avancement intermédiaire au bout de 10h environ
- Démonstration lors de la dernière séance

Remarques sur l'interface graphique

La librairie graphique 2D à privilégier est la *SDL*, pour laquelle on trouve de très bons tutoriels (*openclassrooms* particulièrement). Cependant, si vous connaissez bien une autre librairie graphique au moins équivalente en fonctionnalités, vous pouvez l'utiliser.

Dans le pire des cas, si vous êtes un peu «trop justes» pour développer une interface graphique, vous devrez au moins permettre de visualiser votre plan et votre robot en mode «texte».

Exemple de visualisation en mode «texte» : les murs sont marqués par des "x", la porte de sortie par un "S", le robot par un "R".

[illegible]