amU Faculté des sciences Aix Marseille Université

Année 2024-25

Domaine Sciences et Technologies
Master informatique
Génie logiciel : TD 1
Codes UE : SINA09EL
Mars Rover

1 Spécifications simulateur Mars Rover

Le but de ce TD est de concevoir un simulateur pour des *rovers* de la NASA explorant un terrain. Un terrain marsien doit être parcouru par les *rovers* afin que leurs caméras embarquées puissent obtenir une vue complète du terrain environnant et la transmettre à la Terre. Le terrain est modélisé sous la forme d'une grille qui peut être rectangulaire, toroïdale ou sphérique (détails ci-après). La grille a une largeur l et une hauteur h quel que soit son type. La position d'un *rover* est représentée par un triplet composé des deux coordonnées entières $0 \leq x < l$ et $0 \leq y < h$ et d'une orientation correspondant à un des quatre points cardinaux (Nord, Sud, Est ou Ouest).

Pour contrôler un *rover*, la NASA donne une suite finie de commandes. Les commandes possibles sont :

- *tourner à gauche* : fait tourner le *rover* de 90 degrés vers la gauche (sens inverse des aiguilles d'une montre) ;
- *tourner à droite* : font tourner le *rover* de 90 degrés vers la droite (sens des aiguilles d'une montre) ;
- *avancer* : indique au *rover* d'avancer d'une case dans le sens de son orientation.

L'entrée du problème pourra décrire plusieurs *rovers* en donnant pour chacun les coordonnées initiales de celui-ci (coordonnées d'atterrissage) ainsi que son orientation. Chaque *rover* agit indépendamment l'un après l'autre. On supposera qu'il n'y pas d'interactions entre les *rover* (deux *rovers* peuvent être aux mêmes coordonnées au même moment). Le but étant d'explorer le terrain, la sortie contiendra le pourcentage de coordonnées explorées par les *rovers* (une case étant considérée comme étant explorée si elle correspond à une case dans la trajectoire d'un des *rovers*, y compris leurs coordonnées de départ) ainsi que la position (coordonnées et orientation) finale de chaque *rover* (position à la fin de l'exécution des commandes ou position juste avant la destruction du *rover* si celui-ci sort du plateau).

1.1 Grille rectangulaire

Une grille rectangulaire correspond à la grille de base représentant un rectangle. Les coordonnées $(0, 0)$ correspondent au coin sud-ouest de la grille alors que les coordonnées $(l - 1, h - 1)$ correspondent au coin nord-est de la grille (l étant la largeur de la grille et h sa hauteur). On suppose que tout mouvement faisant sortir le *rover* de la grille rectangulaire entraîne sa destruction et qu'il n'exécute pas les commandes suivantes.

1.2 Grille toroïdale

Une grille toroïdale correspond à une topologie ou le haut et le bas de la grille sont connectés et un robot sortant par le haut réapparait en bas de la grille (passant d'une coordonnée en y de $h - 1$ à 0 avec h la hauteur de la grille et en gardant la même coordonnée en x) et inversement. De même, la gauche et la droite sont connectées et un robot sortant par la gauche réapparait à la droite de la grille (passant d'une coordonnée en x de 0 à $l - 1$ avec l la largeur de la grille et en gardant la même coordonnée en y) et inversement.

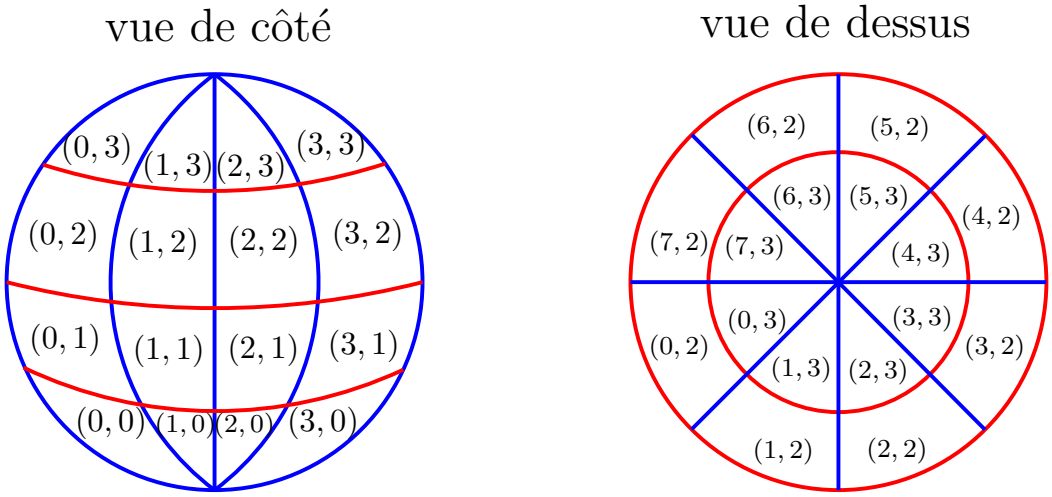
Le tableau suivant donne pour une grille 4×4 ($x \in \{0, 1, 2, 3\}$, $y \in \{0, 1, 2, 3\}$) la position après un mouvement dans la grille :

Position\Opération	NORTH	SOUTH	EAST	WEST
(0, 0)	(1, 0)	(3, 0)	(0, 1)	(0, 3)
(1, 0)	(2, 0)	(0, 0)	(1, 1)	(1, 3)
(1, 1)	(2, 1)	(0, 1)	(1, 2)	(1, 0)
(2, 0)	(3, 0)	(1, 0)	(2, 1)	(2, 3)

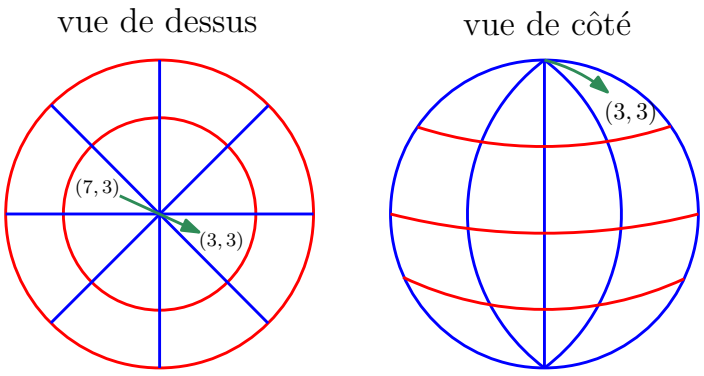
1.3 Grille sphérique

On souhaite définir une grille correspondant à une sphère. Pour cela, on définit un système de latitude et de longitude. La sphère est découpée en un nombre pair de parallèles (cercles horizontaux correspondant à l'intersection de la sphère avec un plan orthogonal à l'axe Nord-Sud) et de méridiens (demi-ellipses reliant le pôle Nord au pôle Sud). Dans ce modèle, x et y deviennent respectivement des représentations abstraites des longitudes et latitudes.

La figure ci-dessous illustre le concept de longitudes et latitudes sont la forme de couples (longitude, latitude) ainsi que les parallèles (en rouge) et les méridiens (en bleu) pour une grille 8×4 ($x \in \{0, 1, 2, 3, 4, 5, 6, 7\}$, $y \in \{0, 1, 2, 3\}$).



Pour la plupart des cas, les mouvements sont définis assez simplement, car la plupart des régions définies par ce système de coordonnées ont quatre côtés et donc quatre régions adjacentes (une par direction possible). Par exemple, la région $(0, 2)$ représentée dans la figure ci-dessus a quatre voisines, $(0, 3)$ au Nord, $(1, 2)$ à l'est, $(0, 1)$ au Sud et $(7, 2)$ à l'Ouest. Les régions touchant un des pôles n'ont que trois côtés et dans ce cas on considère que la région voisine dans la direction du pôle est celle opposé par rapport au pôle. Par exemple, la région considérée comme étant au Nord de la région $(7, 3)$ est la région $(3, 3)$. On considère que le robot traverse le pôle et se retrouve donc de l'autre côté. La région $(7, 3)$ a donc bien quatre voisines, $(3, 3)$ au Nord, $(0, 3)$ à l'Est, $(7, 2)$ au Sud et $(6, 3)$ à l'Ouest. La figure ci-dessous illustre le mouvement en vert du robot de la région $(7, 3)$ à la région $(3, 3)$.



Dans le cas d'un mouvement passant par un pôle, l'orientation du robot est inversée (un robot orienté vers le Nord traversant le pôle Nord devenant orienté vers le Sud et inversement un robot orienté vers le Sud traversant le pôle Sud devenant orienté vers le Nord).

2 Conception et diagramme de classes

Question 1 : Donner un diagramme de classes du simulateur de mars *rover* avec les trois types de grilles.

On souhaite maintenant rajouter une nouvelle commande en plus de trois déjà présentes qui permettrait au *rover* de reculer, c'est-à-dire de se déplacer dans la direction inverse de son orientation. Par exemple un *rover* orienté vers le nord qui recule, se déplace vers le sud tout en gardant son orientation vers le nord.

Question 2 : Modifier le diagramme de classes de la question précédente afin de prendre en compte le fait que les *rovers* puissent reculer.

On souhaite maintenant prendre en compte la présence d'obstacles dans les terrains. Des obstacles (présents sur certaines cases) détruisent tout *rover* s'y déplaçant (y compris pour l'atterrissage si l'obstacle est présent sur ses coordonnées initiales). La sortie du simulateur devra dorénavant indiquer si le *rover* est détruit ou pas à la fin de la simulation.

Question 3 : Modifier le diagramme de classes de la question précédente afin de prendre en compte le fait que des obstacles soient présents sur le terrain.

3 Rappels diagramme de classes