Simon Basilien

UQAC
Université du Québec
à Chicoutimi

Marche Aléatoire

Le 09 Avril 2017

LE FONCTIONNEMENT :

Pour faire fonctionner le code, il n'y a qu'à exécuter le fichier python modele.py, et utiliser l'interface graphique fournie.

Pour le code dans le cadre de cette marche aléatoire, j'ai repris le générateur de nombre aléatoires à récursivité linéaire développé dans le cadre du précédent devoir. Les valeurs utilisées pour l'initialisation de ce dernier ayant été fixée, la même séquence de génération de marche aléatoire, en partant du lancement du script, donnera toujours lieu aux mêmes chemins.

Le code est divisé en deux parties : l'interface graphique, qui s'occupera de la boucle correspondant au nombre de pas, et une classe nommée "walker" qui correspond à l'élément se déplaçant.

Ce "marcheur" est au courant du chemin qu'il a parcouru, de sa position dans l'espace, du nombre de pas effectuée, et des chemins non fructueux trouvés lors de sa balade - dans le cas de l'évitement de soi-même.
Ce marcheur est à même de déterminer où il peut aller, avancer, et reculer (si il atteint un chemin qui ne lui permet plus d'avancer dans le cas de l'évitement de soi-même.
Le chemin est enregistré sous la forme d'un entier. Le codage de cet entier en base 4 nous permet de déterminer le chemin emprunté par le marcheur. Les 4 valeurs possibles correspondent aux directions suivantes :

- 0 : bas
- 1 : gauche
- 2 : haut
- 3 : droite

Ces valeurs ont été choisies de manière arbitraire afin de faciliter certains calculs.

L'interface graphique est responsable de la boucle faisant avancer le marcheur, ainsi que de l'affichage. Elle ne fait que demander au marcheur de se mettre à jour, et effectue les modifications nécessaires au niveau graphique. Si le marcheur fait un retour en arrière, dans le cadre de l'évitement de soi-même, la figure est régénérée de 0, tandis que seul le dernier trajet est ajouté, si il ne fait qu'avancer.
L'interface nous permet aussi de récupérer le nombre de pas désiré par l'utilisateur, ainsi que le type de trajet souhaité. Il est nécessaire de noter que le temps nécessaire pour compléter la marche est proportionnel au nombre de pas lors du trajet simple ou sans retour en arrière, mais bien plus lent dans le cas de l'évitement de soi-même.

Un nombre de pas trop important (plus d'une cinquantaine) dans le cadre de l'évitement de soi-même peut engendrer un blocage au niveau du programme, jusqu'à ce qu'un trajet adéquat soit trouvé.

ETUDE PAR SIMULATION :

Afin d'obtenir le graphique 7, il serait possible d'effectuer des batteries de test avec les différents type de trajet, ou bien de compter la distance au centre pour tous les trajets correspondants. La première possibilité nous donnerait une estimation, et la seconde nous donnerait une valeur exacte, mais demanderait d'effectuer un nombre important de calculs. De même, dans le cadre de l'évitement de soi-même, une grande place en mémoire serait nécessaire.
Il serait aussi nécessaire de créer une seconde interface graphique dans le cadre de ces tests.

Je n'ai malheureusement pas été à même de mettre en place et modèle avec interface pour réaliser ce graphique.