INF304

Mini-projet «Curiosity Revolutions»

Shaghayegh HAJMOHAMMADKASHI , Olinda SELMOOGLU– MIN3

TP7 – Tests de programmes robots

Le code de curiosity-test est organisé de manière modulaire avec des fonctions dédiées à la gestion des erreurs liées au terrain et au programme. La boucle principale exécute le programme du robot tout en surveillant d'éventuelles erreurs et en vérifiant le résultat final. Des commentaires clairs accompagnent le code pour faciliter la compréhension. L'objectif global est de simuler et de valider le comportement du robot dans un environnement défini en fonction d'un programme donné, tout en assurant une gestion robuste des erreurs.
Pour le tester on a effectué les tests suivants:

-Un test de lecture où le fichier terrain n'existe pas (test0_interprete.txt)
-Un test de lecture où le fichier programme n'existe pas(test8_interprete.txt)
-Un test où le robot est tombé dans l'eau(test7_interprete.txt)
-Un test où le robot est sorti du terain(test4_interprete.txt)
- Un test où le robot rencontre un obstacle comme prévu(test3_interprete.txt) et un test où le robot s'écrase contre un obstacle mais ce n'était pas le résultat attendu(test5_interprete.txt).
-Des tests où le programme est termine en différents orientations (test1_interprete.txt),(test2_interprete.txt),(test6_interprete.txt)

TP8 – Génération aléatoire de terrains

Le fichier "test_generation_terrain.c" constitue un programme de test dédié à la génération aléatoire de terrains pour un système robotique. Paramétré par des arguments en ligne de commande tels que le nombre de terrains à créer (N), la largeur, la hauteur, et la densité d'obstacles souhaitée, le programme utilise la fonction generation_aleatoire pour générer des terrains aux caractéristiques spécifiées.
Une attention particulière est portée à la validation des terrains, assurant qu'un chemin praticable vers la sortie est présent. Les détails de chaque terrain, comme le nombre réel d'obstacles, la densité effective d'obstacles, et la densité attendue, sont consignés à la fois dans la console et dans un fichier de résultat défini par l'utilisateur. Des mécanismes sont intégrés pour éviter la répétition de terrains identiques, introduisant une pause entre les générations successives.
En conclusion, ce programme fournit des statistiques globales, telles que le nombre total de terrains générés, le pourcentage de terrains valides par rapport au total généré, et la densité moyenne des obstacles dans les terrains obtenus, contribuant ainsi à une évaluation approfondie de la génération aléatoire de terrains dans un contexte robotique.

Pour le tester on a.

Le Programme 1 et le Programme 2 représentent des améliorations significatives par rapport au simple programme "Avancer indéfiniment". Ces deux programmes intègrent des mécanismes sophistiqués pour prendre des décisions en fonction de la disponibilité de la case devant le robot, permettant ainsi d'ajuster sa trajectoire en évitant les obstacles.

Le Programme 1 utilise la mesure de la case devant le robot pour tourner à droite en cas de disponibilité et avancer autrement. De même, le Programme 2 tourne à gauche en présence d'une case libre et avance autrement. Ces adaptations intelligentes améliorent la capacité du robot à naviguer dans des terrains complexes et à trouver un chemin vers la sortie.

Les résultats statistiques obtenus pour le Programme 1 et le Programme 2 dans différentes configurations de terrains montrent des performances variables. Dans certaines configurations, les deux programmes parviennent presque toujours à sortir du terrain, tandis que dans d'autres, ils peuvent rencontrer des blocages. Ces résultats mettent en évidence l'importance de la configuration du terrain et de la densité d'obstacles dans l'efficacité des programmes.

En conclusion, les améliorations apportées par le Programme 1 et le Programme 2, grâce à leur capacité à prendre des décisions éclairées basées sur la mesure de la case devant le robot, les rendent plus adaptés à la résolution de terrains complexes avec des obstacles. Les résultats statistiques fournissent des insights précieux sur la performance de ces programmes dans des conditions variées, soulignant la nécessité de considérer la diversité des terrains pour évaluer leur efficacité.

Les terrains construits ainsi que les RÉSULTATS STATISTIQUES des performances du Programme 1 et du Programme 2 ont été consignés de manière détaillée dans les fichiers de résultat, accessibles sous le répertoire "fichier_res/terr_resultat*.txt".

TP9 – Observateurs et verification dynamique

Le fichier "observateur.c" qu'on a fourni est pour spécifier le paquetage observateur basé sur l'automate. Voici une explication des éléments clés de ce fichier :

- Définition des Caractères : La définition d'une énumération Caractere permet de représenter les différentes actions possibles que l'observateur peut prendre. Les actions incluent Avancer (A), Tourner à gauche (G), Tourner à droite (D), et Mesurer (M).
- Définition des États : L'énumération Etat représente les différents états possibles de l'automate observateur. Les états comprennent INITIAL, MESURE et ERREUR.
- Fonction initial : Cette fonction retourne l'état initial de l'automate, qui est INITIAL dans ce cas.
- Fonction transition : Cette fonction prend en paramètre l'état actuel et le caractère en entrée, puis retourne le nouvel état résultant de la transition. La logique de transition est définie dans cette fonction en utilisant une structure de commutation (switch).
- Fonction est_accepteur : Cette fonction prend en paramètre un état et retourne 1 si l'état est accepteur (différent de ERREUR), et 0 sinon. Elle permet de déterminer si l'observateur a atteint un état valide.

Conclusion

En conclusion, ce projet a abouti à la mise en œuvre d'une méthode d'interprétation de commandes à partir de fichiers texte fournis en entrée standard. Ces interpréteurs intègrent des mécanismes de validation pour détecter les erreurs et fournir des retours pertinents sur les résultats obtenus après l'interprétation. Afin de minimiser les risques d'erreurs inattendues et d'automatiser l'utilisation du projet, un programme a été développé pour générer des environnements terrains aléatoires.

Ce projet a mis en lumière l'importance cruciale de la communication efficace, d'un code bien structuré et commenté, ainsi que d'une coordination, planification et organisation adéquates entre les membres de l'équipe. Ces aspects se sont avérés essentiels pour créer un projet complet qui fonctionne comme prévu, tout en minimisant les erreurs potentielles. La réussite de ce projet a démontré l'importance de la collaboration et de la gestion efficace des ressources pour atteindre les objectifs fixés.