amU Faculté des sciences Aix Marseille Université

Année 2024-25

Domaine Sciences et Technologies
Master informatique
Génie logiciel : TP 3
Codes UE : SINA09EL
Mars Rover (3/3)

1 Fonctionnalités optionnelles

Si vous avez fini d'implémenter les fonctionnalités décrites dans les deux premières planches de TP, vous pouvez ajouter des fonctionnalités supplémentaires au simulateur *mars-rover*, comme les suivantes :

- **Ajout d'obstacles sur le terrain :** On considère qu'il y a maintenant 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). Le format de fichier d'entrée doit être modifié afin de pouvoir rajouter une liste de coordonnées correspondant aux cases contenant des obstacles. La sortie du simulateur devra dorénavant indiquer si le *rover* est détruit ou pas à la fin de la simulation.
- **Capacité d'exploration augmentée pour les *rovers* :** Chaque *rover* a désormais un rayon d'exploration qui lui permet d'explorer à tout moment lors de sa trajectoire toutes les cases qui sont à une distance de sa case inférieure ou égale à son rayon d'exploration. Un rayon d'exploration de zéro correspond à la configuration des TP précédents. Un rayon d'exploration égal à un permet au *rover* d'explorer toutes les cases voisines à une des cases de sa trajectoire. Un rayon d'exploration égal à deux permet d'explorer les case à distance deux (voisine des voisines).
- **Interface graphique pour la visualisation :** Le but est d'ajouter une interface permettant de visualiser les trajectoires d'exploration par les *rovers*. Le terrain sera représenté par une grille, l'affichage des cases devant permettre de distinguer si la case est explorée et/ou si elle contient un ou plusieurs *rovers*. L'interface devra permettre de charger un fichier de configuration puis de lancer la simulation soit pas à pas ou bien via une animation.

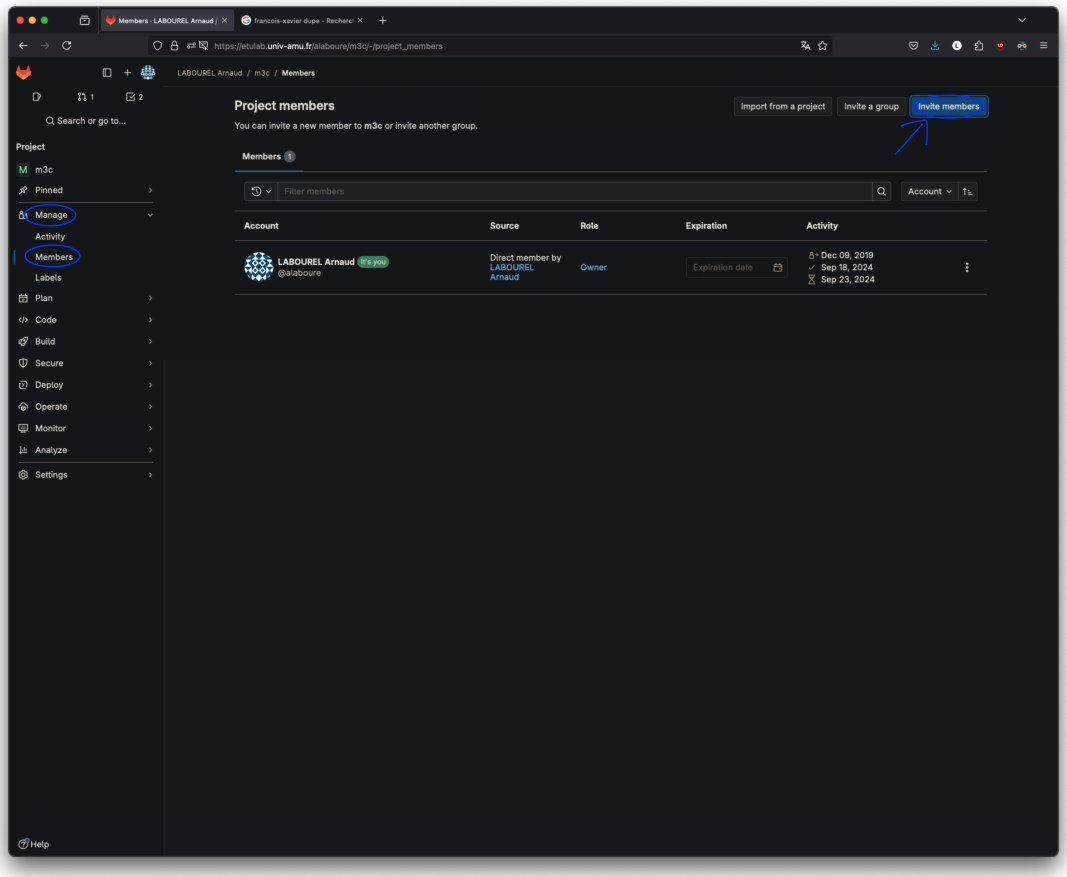
2 Consignes pour le rendu

2.1 Rendu via `etulab`

Le rendu de votre projet *mars-rover* est à faire pour le 3 octobre 2024. Il faudra que le dépôt *git* sur `etulab` soit à jour pour le 3 octobre 2024 à 23h59 et que les trois enseignants de l'UE (liste ci-dessous) soient membres du projet avec des droits au moins égal à `reporter`.

- Arnaud Labourel, identifiant `alaboure`
- Jean-Luc Massat, identifiant `massat`
- François-Xavier Dupé, identifiant `fdupe`

L'ajout de membre se fait via *manage* → *members* dans le menu de votre projet sur *etulab*, puis en cliquant sur le bouton *invite members*.



Le travail peut être réalisé seul ou en binôme. Dans le cas de binôme, les deux étudiants devront être membres du projet.

2.2 Respect de la propriété intellectuelle

Comme pour tout devoir, nous vous demandons de ne pas partager votre programme, complet ou partiel, avec des étudiants n'étant pas membres de votre projet. Tout emprunt que vous effectuez doit être proprement documenté en indiquant quelle partie de votre programme est concernée et de quelle source elle provient (nom d'un autre étudiant, site internet, ...). Les emprunts incluent l'utilisation d'IA génératives telles que ChatGPT qui devront donc aussi être documentés.

2.3 Fichier `README.md` du projet

Votre projet devra contenir à sa racine un fichier `README.md` contenant les informations sur ces membres et les emprunts réalisés. Le format du fichier devra être le suivant :

```
# Mars rover

## Membres du projet

- NOM Prénom du premier membre
```

```
- NOM Prénom du deuxième membre (si applicable)

## Description des emprunts

- Utilisation de ChatGPT pour les classes : `Main.java`, ...
- ...
```

2.4 Critères d'évaluation

Vous serez évalué sur :

— **La conception logicielle :** votre projet devra dans la mesure du possible respecter les bonnes pratiques de conception logicielle en programmation orientée objet tels que les principes SOLID. Par exemple, des classes ayant trop de responsabilités vous pénaliseront.

— **La propreté du code :** comme indiqué dans le cours, il est important de programmer proprement. Des répétitions de code trop visibles, des noms mal choisis ou des fonctions ayant beaucoup de lignes de code (plus de dix) vous pénaliseront. Le sujet vous donne les méthodes que vous devez absolument écrire, mais il est tout à fait autoriser d'écrire des méthodes supplémentaires, de créer des constantes, ... pour augmenter la lisibilité du code. On rappelle que vous devez écrire le code en anglais.

— **La correction du code :** on s'attend à ce que votre code soit correct, c'est-à-dire qu'il respecte les spécifications dans le sujet. Comme indiqué dans le sujet, vous devez tester votre code pour vérifier son comportement.

— **Les commit/push effectués :** il vous faudra travailler en continu avec `git` et faire des *push/commit* le plus régulièrement possible. Un projet ayant très peu de *push/commit* effectués juste avant la date limite sera considéré comme suspicieux et noté en conséquence. Un minimum accepté pour le projet sera d'au moins **2 pushs sur deux jours différents** et d'au moins **10 commits** au total. Dans le cas d'un projet réalisé en binôme, chacun des deux membres du projet devra réaliser un *push*.