Documentation : Structure MVC dans le Projet de Simulation de Robot

Introduction

Cette documentation explique la structure **MVC (Modèle-Vue-Contrôleur)** utilisée dans le projet de simulation de robot, basée sur l'architecture du projet visible dans l'explorateur de fichiers.

1. Qu'est-ce que l'Architecture MVC ?

L'architecture MVC est un modèle de conception logicielle qui sépare l'application en trois composantes :

- **Modèle** : Gère les données et la logique de l'application.
- **Vue** : Gère l'affichage et l'interface utilisateur.
- **Contrôleur** : Sert d'intermédiaire entre le modèle et la vue, gère les entrées de l'utilisateur.

2. Analyse de la Structure du Projet

1. Dossier `controller/`

Contient les contrôleurs qui gèrent la logique de l'application et les interactions :

- `map_controller.py` : Contrôle les interactions avec la carte.
- `robot_controller.py` : Gère les mouvements et l'état du robot.
- `simulation_controller.py` : Coordonne l'ensemble de la simulation.

2. Dossier `model/`

Gère les données et la logique métier du projet :

- `map_model.py` : Contient les données liées à l'environnement et à la carte.

- `robot.py` : Contient les attributs et fonctions du robot, notamment les calculs de position et de vitesse.

3. Dossier `view/`

Contient les éléments visuels et l'affichage de la simulation :

- `app_view.py` : Vue principale de l'application.
- `control_panel.py` : Interface pour contrôler la simulation.
- `map_view.py` : Affichage de la carte.
- `robot_view.py` : Affichage du robot et de ses mouvements.

3. Fonctionnement de l'Architecture MVC dans le Projet

- **Le modèle (`model/`)** contient les données du robot et de l'environnement.
- **Le contrôleur (`controller/`)** récupère les commandes de l'utilisateur, met à jour le modèle et informe la vue des changements.
- **La vue (`view/`)** affiche l'état actuel du modèle et reçoit les interactions utilisateur.

Exemple : Lorsque l'utilisateur modifie la vitesse du robot via l'interface (`view`), le `robot_controller.py` ajuste les vitesses, met à jour les positions dans `robot.py` (modèle) et notifie `robot_view.py` pour refléter les changements.

4. Avantages de l'Architecture MVC

- **Séparation claire des responsabilités** : Facilite la maintenance et l'évolution.
- **Réutilisation** : Les vues peuvent être modifiées sans impacter la logique métier.
- **Testabilité** : Chaque composant peut être testé indépendamment.

5. Conclusion

L'architecture MVC rend le projet de simulation modulaire, lisible et extensible. Chaque couche a un rôle bien défini, permettant une gestion fluide des mouvements du robot et de l'affichage.