PROJET ROBOTIQUE

ROBOTECH

Sommaire

- Introduction du projet
- Description global de notre code
- Choix de conception
- Fonctionnalités du robot
- Stratégies disponibles
- Robot Réel
- Conclusion

Introduction:
But du projet

- Implémenter une simulation pour notre robot en suivant le design pattern MVC et lui faire des tâches.

Architecture du code

Dans le dossier RoboTech nous retrouvons les fichiers et dossiers :

projet.py

camera_test: fichiers de test pour la caméra

module :

- **controleur :**
  - **controleur.py**
- **modele :**
  - **element_simulation.py**
  - **proxy.py**
  - **robot_api.py**
  - **robot_mock_up.py**
- **vue :**
  - **affichage_2D.py**
- **camera.py**
- **constante.py**
- **core.py**

test : contient des fichiers de tests pour nos modules

Compte-rendu des séances : contient tous les comptes-rendus réalisés

Tests

Nous avons importé le module unittest de la bibliothèque standard de Python qui inclut le mécanisme des tests unitaires.

Pour cela nous définissons une classe héritant de unittestTestCase, et nous définissions ensuite une méthode dont le nom commence par test.

Pour exécuter tous les tests unitaires on exécute la commande : **python -m unittest discover test -v**

Choix de conception:
Robot, Objet et Environnement

- Environnement sous la forme d'un terrain continu afin d'obtenir une plus grande precision.
- Le robot et les objets de l’environnement, on a opté pour une représentation sous forme de cercle.
- Utilisation des coordonnées réelles pour représenter la position du robot et des objets afin d’assurer une précision optimale dans un environnement continu.

Interface graphique

- Utilisation de Tkinter

- Canevas en fonction des valeurs de largeur, hauteur et échelle de l'environnement, on utilise également un multiplicateur « mult » défini dans le fichier « constante.py » pour multiplier tous les éléments présents sur le canevas.

- Représentation graphique du robot: cercle rouge pour le représenter avec la fonction « create_cercle »

Gestion du temps

- Pour gérer le temps dans le code, on a utilisé une fonction « sleep » dans le thread du contrôleur, en lui passant un pas de temps « dt » défini dans le fichier « constante.py »

- Pour calculer le temps écoulé, on fait la différence entre le temps actuel obtenu grâce à la fonction « time.time() » et le temps de la dernière mise à jour. Ce calcul est nécessaire pour les calculs d'angle, de distance et de position.

- En ralentissant la mise à jour du contrôleur par rapport à la simulation, on permet à la simulation de s'exécuter plus fréquemment

Fonctionnalités du robot

- Deux roues motrices, une gauche et une droite, un capteur de distance et une caméra.
- Les roues sont contrôlables séparément.
- La caméra peut prendre des photos, ce qui peut permettre entre autres de reconnaître un obstacle ou suivre un chemin.

Stratégies disponibles

- **StrategieAvance**
- **StrategieAngle**
- **StrategieArretMur**
- **StrategieSeq**

Robot réel

Conclusion