Semestre Février 2021

RAPPORT DE PROJET

LU2IN013 Groupe 3

Projet Robotique

Groupe Road to 100/100

Christian Zhuang
Maxime Millard
Youheng Guo
Jeremy Boda
Simon Gripon
André Vicente

Sommaire :

I. Présentation du Projet

Dans l'ue LU2IN013, notre objectif principal est d'apprendre à faire un projet en groupe. Nous avons utilisé principalement deux méthodes - Agile et Scrum - pour organiser le développement. Cela nous a permis d'analyser les besoins du client (ici les consignes pour ce projet) pour optimiser au mieux notre temps et notre efficacité.

Ce projet consiste à faire réaliser des tâches à un robot Gopigo 3, ainsi qu'à faire une simulation d'un robot virtuel qui agit de la même manière que le robot réel. Trois fonctionnalités principales étaient à implémenter : faire avancer le robot selon une certaine séquence afin de lui faire "tracer" un carré, le faire s'approcher au plus près des obstacles sans toutefois les toucher, et enfin le faire suivre une balise possédant des caractéristiques spécifiques.

II. Description de l'architecture du code

A. Modules utilisés

Pour la réalisation de ce projet, nous avons dû utiliser différents modules Python que voici :

- Tkinter : module intégré dans Python pour obtenir une interface graphique 2D permettant d'afficher et interagir avec la simulation.
- Panda3d : un moteur de jeu permettant de créer des modèles 3D qui nous a servi à faire une représentation 3D de la simulation.
- PIL (Pillow) : un module pour le traitement d'image de la caméra du robot.

B. Description des fichiers et des classes réalisées

Fichier	Nom de Classe	Description
gopigo.py	Robot2I013	Classe test du robot réel, qui retourne uniquement des informations sur la réussite de l'appel des fonctions du robot.
robot.py	Robot	Représente un robot virtuel qui reprend les mêmes fonctionnalités (et les simule) que celles du robot réel.
	Robot_Proxy	Classe qui relie le robot réel à la Classe Robot
robot2I013.py	Robot2I013	API du robot reel, utilisée pour lui donner des instructions
arene.py	Arene	Simule le terrain où le robot peut se déplacer en utilisant une matrice à deux dimensions pour représenter une carte où se trouve le robot, des obstacles et la balise.
fenetre.py	Fenetre	- Utilise le module Tkinter - Classe qui gère l'affichage graphique de la simulation et peut envoyer des intentions avec des boutons
	Fram_Cam	Classe qui relie l'affichage graphique de la classe Camera (Panda3d) à la Fenetre (Tkinter).
controler.py	Controler	Classe qui gère toutes les stratégies qui manipulent les mouvements du robot réel et virtuel.
strategy.py	StrategyAvance	Stratégie qui fait avancer le Robot selon une distance donnée sans heurter d'obstacle.
	StrategyTourner	Stratégie qui fait tourner le Robot selon un angle et une direction.
	StrategySequence	Stratégie qui exécute une liste de stratégies dans l'ordre.

	StrategyChercherBalise	Stratégie qui permet de chercher la balise autour du robot.
view3D.py	Camera	- Utilise le module Panda3d - Classe qui donne une vision 3D du Robot dans la simulation
cube.py	CubeMaker	Classe qui génère un cube représentant un obstacle 3D de la simulation.
balise.py		- Utilise le module PIL - Contient les fonctions qui permet de reconnaître la balise sur une image

III. Capacités de la simulation

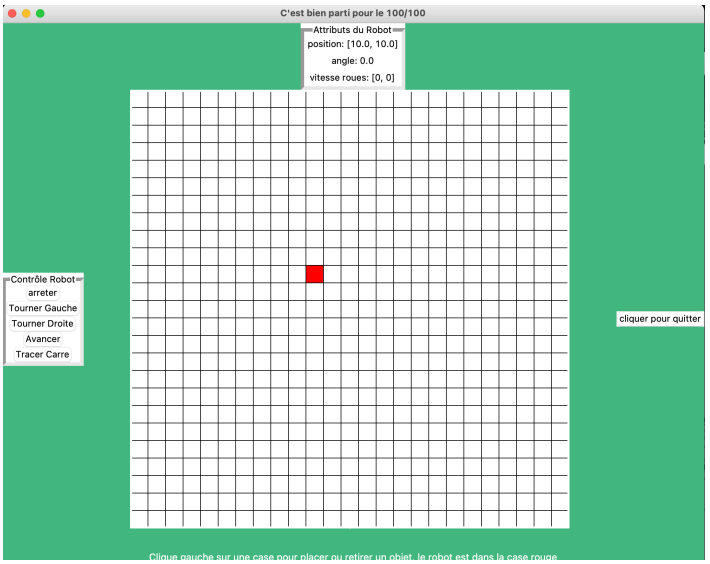
L'affichage de la simulation se décompose en deux parties : une fenêtre de contrôle, et une vue en 3D à la première personne du robot.

La fenêtre de contrôle se compose d'une carte sous forme de matrice et d'une série de boutons. Sur la carte, le robot est représenté en rouge, les obstacles en vert et la balise en violet. On peut ajouter un obstacle en cliquant sur une case vide, et en supprimer un en cliquant dessus. Les boutons activent des fonctionnalités quand on clique dessus, comme faire avancer le robot, le faire tourner, le faire suivre la balise etc. De plus, on retrouve sur cette fenêtre des informations sur la position, l'orientation et la vitesse du robot.

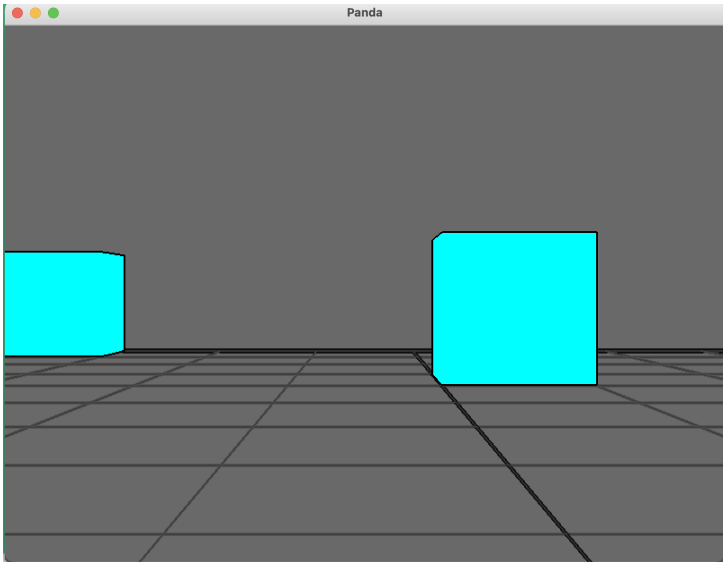
L'affichage avec la vue 3D permet de voir en direct les actions réalisées dans la fenêtre de contrôle, que ce soit le placement des obstacles ou les directives données au robot. Cette vue est l'équivalent de la vision de la caméra du robot réel. Les obstacles y sont représentés en bleu cyan, et la balise par un cube dont une des faces est une représentation de la balise réelle.

Lien vers la démonstration de la simulation :

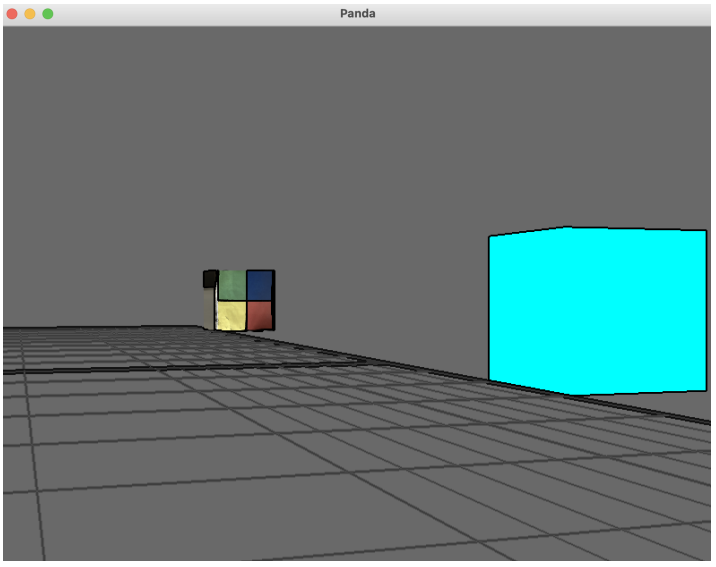
- https://www.youtube.com/watch?v=w4VYcenmtrE



La fenêtre de contrôle avec le robot en rouge



Les obstacles dans la vue 3D



La balise (et un obstacle) dans la vue 3D

IV. Capacités de l’IA

- Lorsque le robot se déplace, il peut approcher jusqu’à 3 millimètres d’un obstacle avant de s'arrêter automatiquement. Il changera de direction avant de reprendre sa route s' il n’a pas fini son chemin.
- Lorsqu’on demande au robot de suivre la balise, il va d’abord utiliser le rotor de sa caméra, ou au besoin effectuer une rotation sur lui-même pour repérer la balise, il va ensuite se diriger dans sa direction, en adaptant son orientation en cas de déplacement de la balise.

V. Initiatives entreprises

Peu d’initiatives ont été prises vis à vis des consignes originelles pour ce qui est des fonctionnalités finales, les consignes originelles représentant pour nous déjà une bonne base à atteindre au vu des difficultés impliquées pour certains d’entre nous par la crise sanitaire, aussi bien d’un point de vue psychologique que scolaire.