SORBONNE UNIVERSITÉ

Projet Robotique

Projet de développement

LU2IN013

Groupe - ITM3

Mouna BAHAMID Meriem BERRAH Ines BOUCHIBANE

Mohamed BOUZEKRI Takoua HADJ ALI

Sommaire

1-Objectif et description du projet

2-Architecture globale de notre projet

3-Gestion du travail en groupe

4-Conclusion

Objectif et description du projet :

Ce projet vise à concevoir un système de contrôle pour un robot GoPiGo, équipé d'un Raspberry Pi et d'une carte Arduino, permettant d'exécuter des stratégies de mouvement, de gérer des distances par rapport à des obstacles, et de suivre une balise via reconnaissance d'image. Les objectifs principaux incluent la simulation du robot dans des environnements 2D et 3D, ainsi que l'application des stratégies sur le robot réel.
Le projet est divisé en trois parties : 2D (simulation avec Pygame) , 3D (simulation avec Panda3D) , et Réel (contrôle du GoPiGo physique). Ce rapport détaille l'architecture, les fonctionnalités, les résultats, et les recommandations pour chaque partie.

- Interface 2D :

Au début du projet, nous avons simulé un robot dans un environnement en deux dimensions. Cette étape nous a permis de comprendre le comportement du robot, ses interactions avec l'environnement, et la mise en œuvre de ses fonctionnalités.

L'objectif était de simuler le robot, son environnement, et les obstacles qu'il pourrait rencontrer, tout en les affichant dans une interface 2D.

Nous avons développé des modules pour le robot, l'environnement, le contrôleur, et les stratégies. Ces composants ont permis de créer une simulation, visualisée via une interface graphique 2D.

Nous avons utilisé Pygame, un module Python adapté pour le rendu graphique simple et interactif. Pygame nous a offert la flexibilité nécessaire pour notre simulation, par rapport à d'autres options comme Tkinter.
Cette interface nous a permis de représenter le robot, de gérer des obstacles (rectangles, cercles), et de tester des stratégies telles qu'avancer, tourner, tracer un carré, ou s'arrêter près d'un obstacle

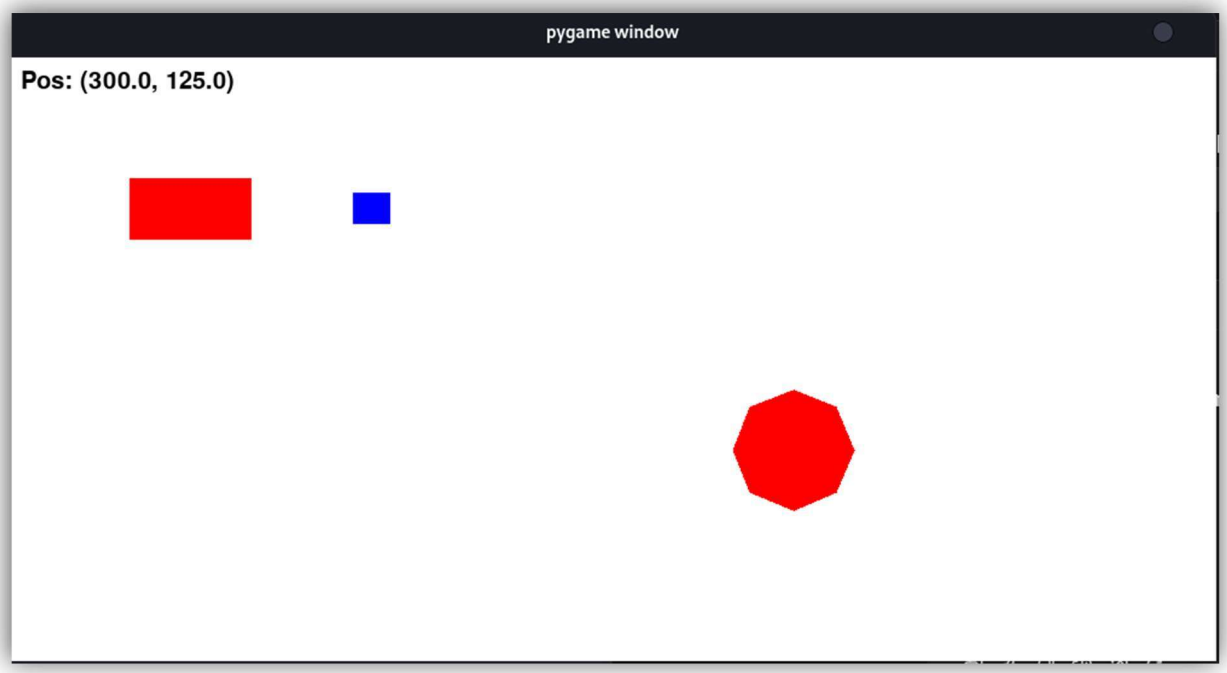


Figure 1: Interface 2D de démonstration

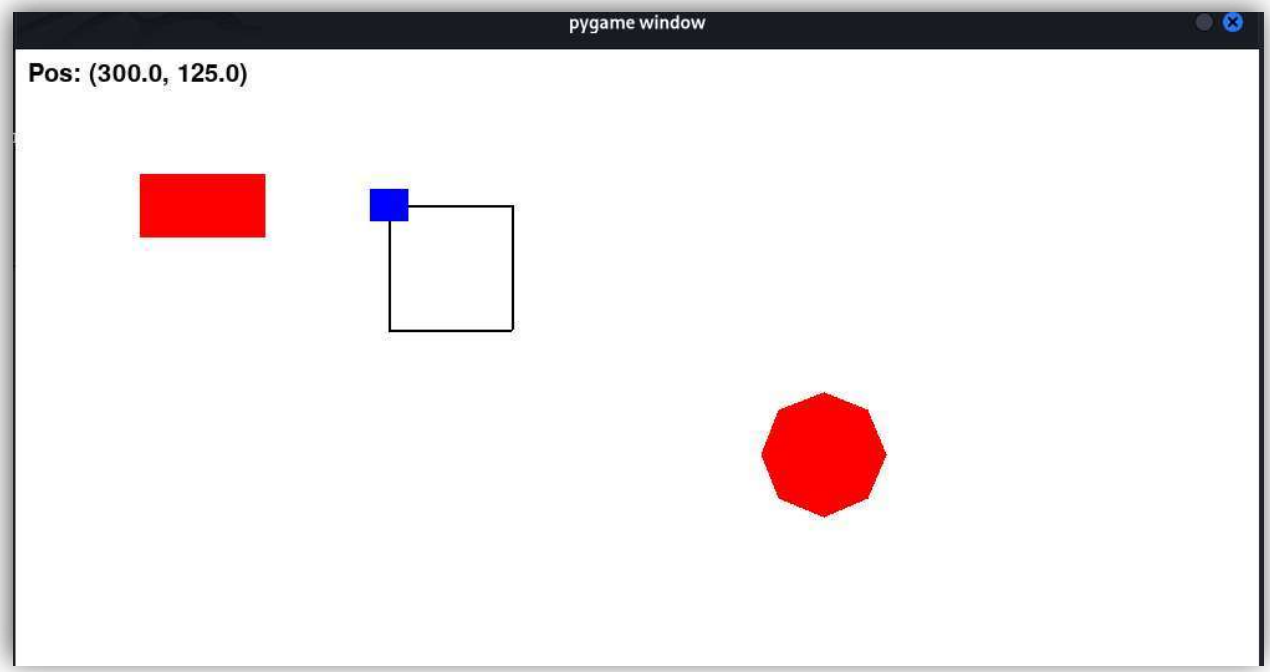
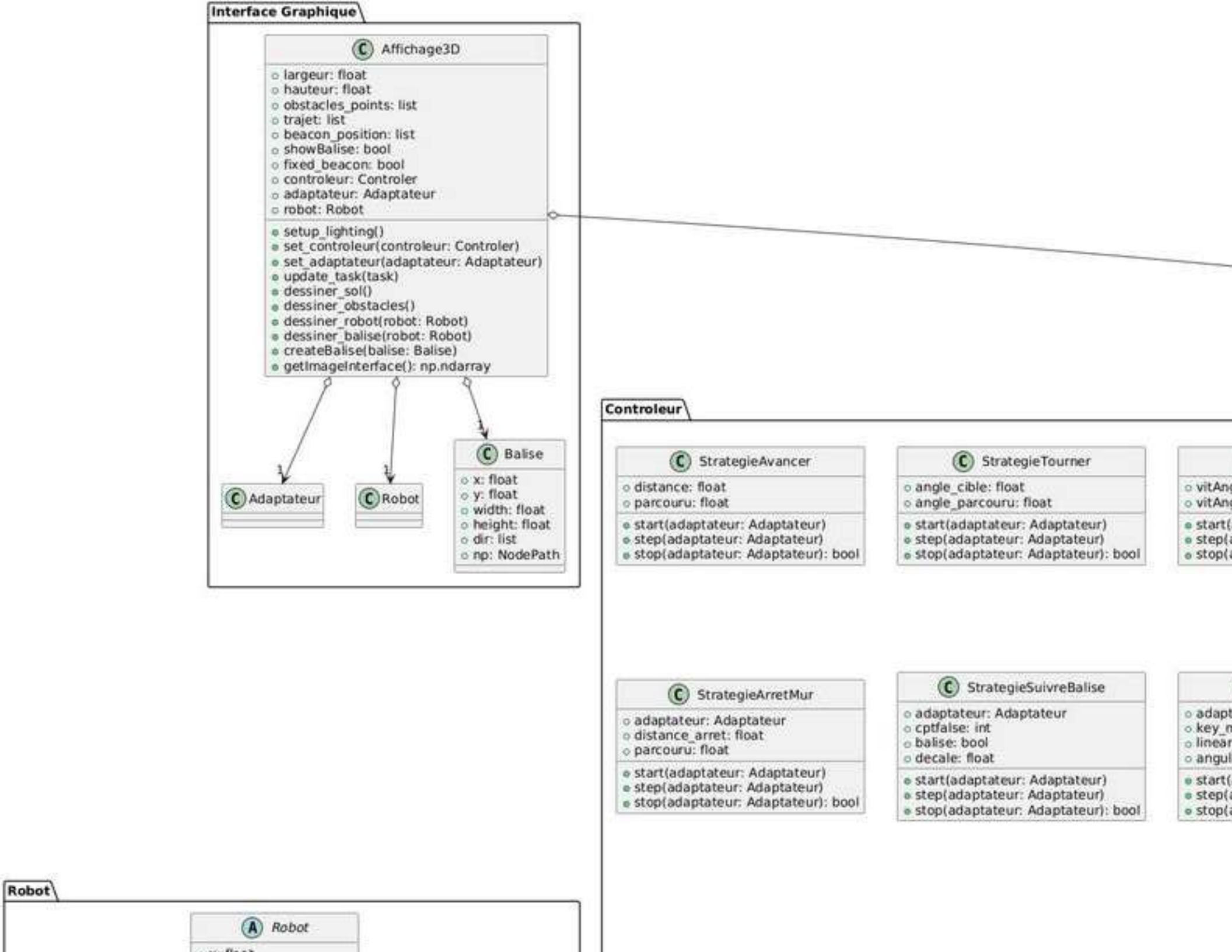


Figure 2: Robot effectuant la stratégie “Tracer carré”


Interface Graphique
Affichage3D
largeur: float
hauteur: float
obstacles_points: list
trajet: list
beacon_position: list
showBalise: bool
fixed_beacon: bool
controleur: Controler
adaptateur: Adaptateur
robot: Robot
setup_lighting()
set_controleur(controleur: Controler)
set_adaptateur(adaptateur: Adaptateur)
update_task(task)
dessiner_sol()
dessiner_obstacles()
dessiner_robot(robot: Robot)
dessiner_balise(robot: Robot)
createBalise(balise: Balise)
getImageInterface(): np.ndarray
Adaptateur
Robot
Balise
x: float
y: float
width: float
height: float
dir: list
np: NodePath
Controleur
StrategieAvancer
distance: float
parcouru: float
start(adaptateur: Adaptateur)
step(adaptateur: Adaptateur)
stop(adaptateur: Adaptateur): bool
StrategieTourner
angle_cible: float
angle_parcouru: float
start(adaptateur: Adaptateur)
step(adaptateur: Adaptateur)
stop(adaptateur: Adaptateur): bool
StrategieArretMur
adaptateur: Adaptateur
distance_arret: float
parcouru: float
start(adaptateur: Adaptateur)
step(adaptateur: Adaptateur)
stop(adaptateur: Adaptateur): bool
StrategieSuivreBalise
adaptateur: Adaptateur
cptfalse: int
balise: bool
decale: float
start(adaptateur: Adaptateur)
step(adaptateur: Adaptateur)
stop(adaptateur: Adaptateur): bool
Robot
Robot

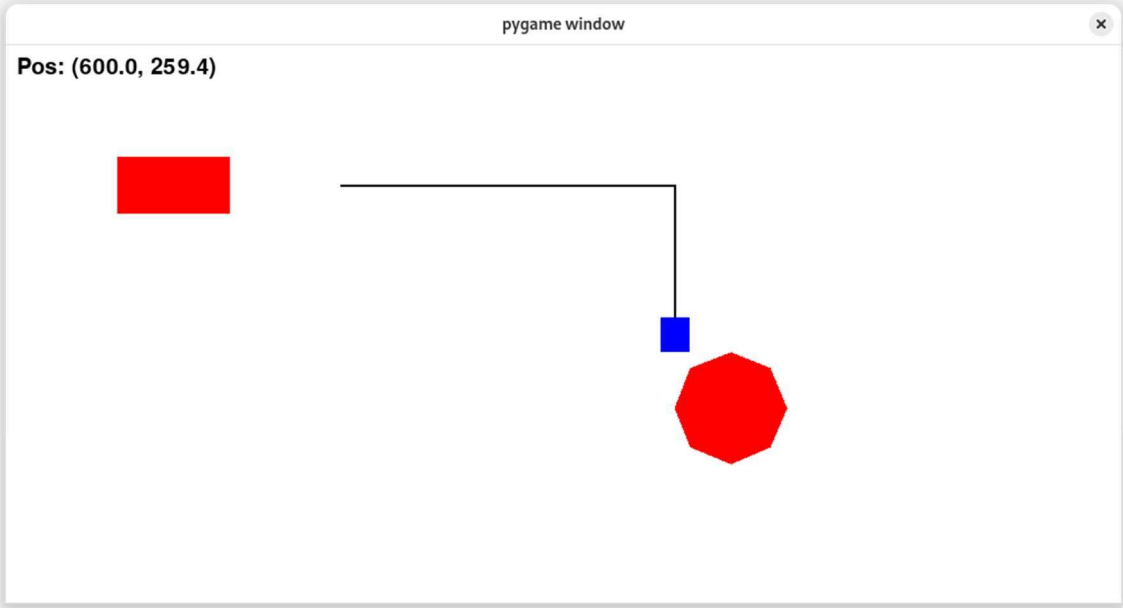


Figure 3: Le robot crashé dans un obstacle

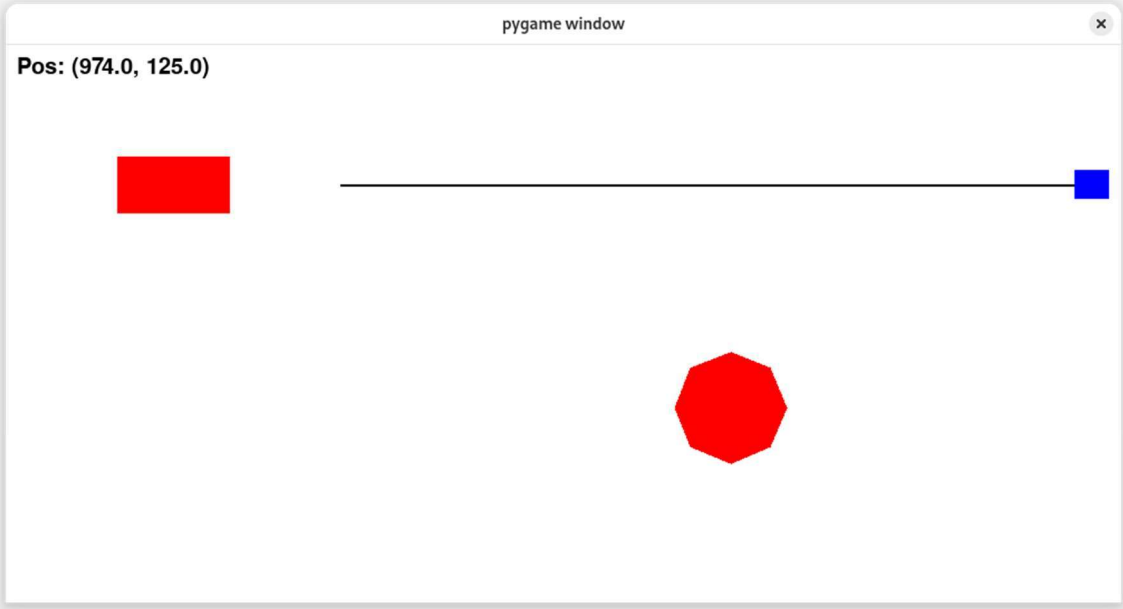


Figure 4: le robot crashé dans la bordure de l'interface

- Interface 3D :

Après la simulation 2D, nous avons développé une interface en trois dimensions pour tester les stratégies dans un environnement plus réaliste. L'objectif était de simuler le robot, son environnement avec des obstacles, et une balise pour la stratégie de suivi, tout en offrant des vues caméras variées.
Nous avons utilisé Panda3D, un module Python puissant pour la création de jeux vidéo 3D, grâce à sa capacité à gérer des scènes complexes et sa documentation riche. Nous avons repris les modules du robot, de l'environnement, du contrôleur, et des stratégies, en les adaptant à l'interface 3D.
Cette interface affiche le robot, des obstacles (rectangles, triangles, cercles), et une balise, avec des vues caméra (dessus, arrière, avant). Elle nous a permis de tester des stratégies comme avancer, tourner, tracer un carré, s'arrêter près d'un obstacle, et suivre une balise, en simulant la caméra du robot.

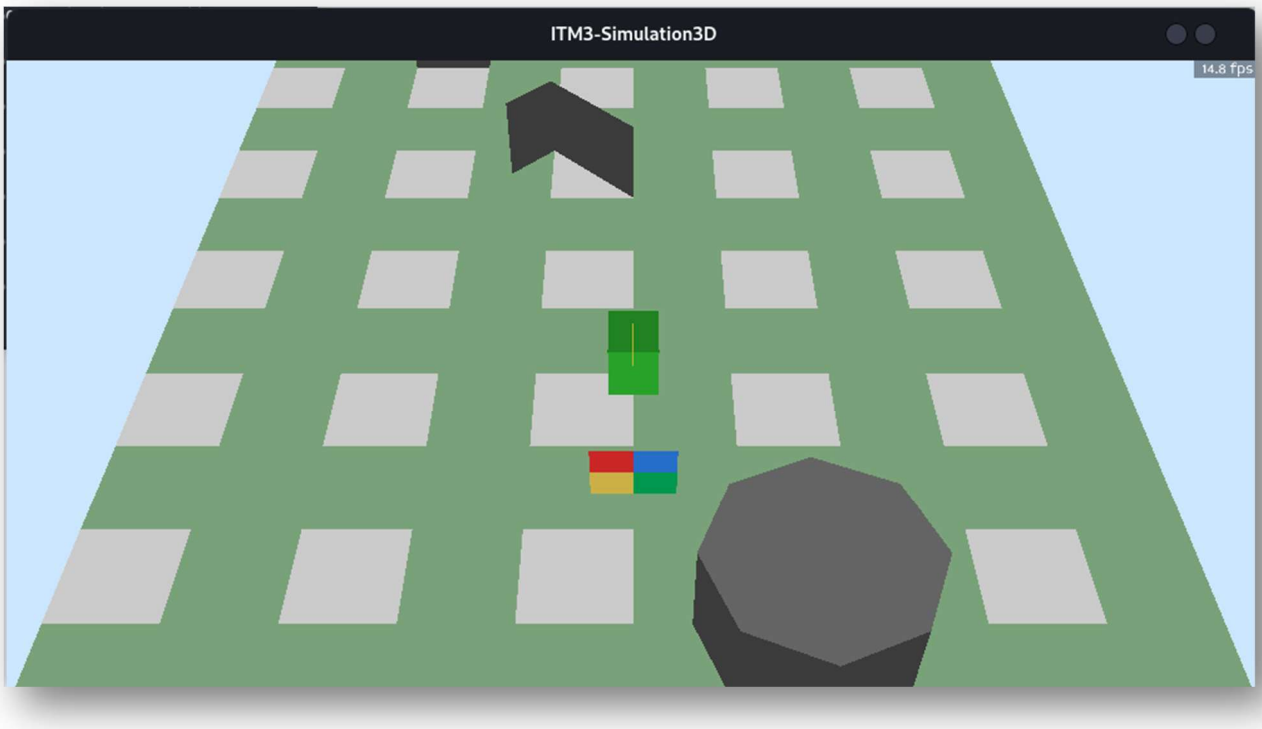


Figure 5: Vue du haut de l'interface 3D

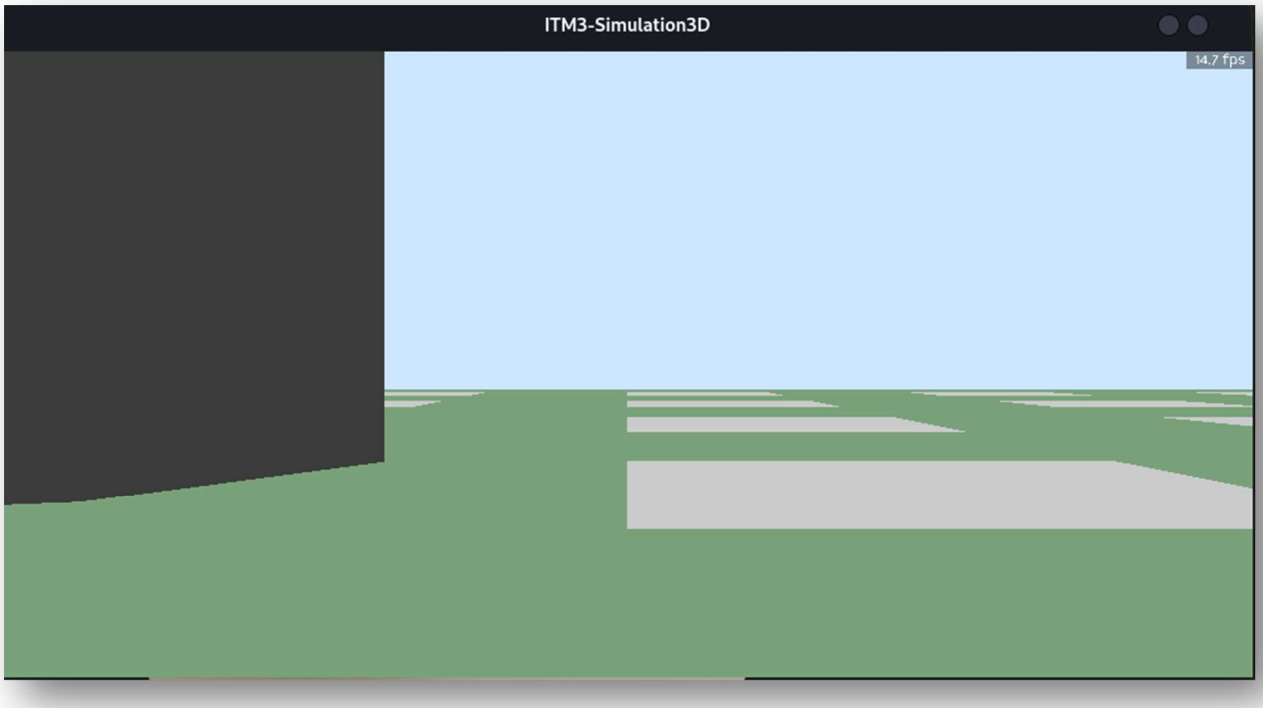


Figure 6: Vue de devant du robot

- Robot Réel :

En parallèle de la simulation, nous avons adapté nos stratégies pour contrôler le robot GoPiGo réel. L'objectif était d'exécuter les stratégies testées en simulation, comme le déplacement libre, tout en tenant compte des contraintes physiques du robot.

Nous avons utilisé des modules fournis pour interfacer les stratégies avec le GoPiGo, en s'appuyant sur la bibliothèque easygopigo3 pour contrôler les moteurs, le capteur de distance, et la caméra (picamera).

Un adaptateur traduit les commandes abstraites en actions physiques, ajustant par exemple le calcul des rotations des roues et les offsets des encodeurs. Pour certaines tâches, nous avons exploité la caméra pour capturer des images, traitées avec cv2 pour analyser l'environnement. Ces adaptations ont permis au robot d'effectuer des actions comme avancer, tracer un carré, ou s'arrêter près d'un mur.

Architecture globale de notre projet

🗁 Description générale :

Le projet se trouve dans le répertoire **projet-robotique-groupe-ITM3**. Il est structuré en trois dossiers et plusieurs fichiers exécutables, en plus des fichiers de test unitaires :

- **NosTaches** : comporte les fiches, les comptes-rendus de chaque séance, ainsi que le diagramme UML du projet.

- **Src :** est un module dans lequel se trouvent plusieurs fichiers dont utils.py. Ce fichier nous permet de définir des fonctions utilitaires et des constantes partagées à travers le projet.
De plus, src contient quatre sous-modules :

  - **controleur** :
    Ce sous-module gère la logique de contrôle du robot en lui donnant des instructions et en définissant des stratégies. Il contient :
    - controleur.py : le contrôleur principal,
    - strategies.py : les différentes stratégies de navigation,
    - adapt.py : classe abstraite pour les adaptateurs,
    - adapt_simule.py et adapt_reel.py : adaptateurs spécifiques pour simulation ou robot réel.

  - **environnement** :
    Ce sous-module définit l'environnement simulé :
    - environnement.py : définition de l'environnement simulé et méthodes associées,
    - obstacle.py : classe Obstacle permettant de définir les obstacles présents dans les simulations.

  - **interface_graphique** :
    Ce sous-module contient deux sous-dossiers :
    - interface2D/ : définit l'interface 2D avec interface2d.py et menu2d.py.

- interface3D/ : définit l'interface 3D

avec interface3d.py, menu3d.py et un dossier assets/ contenant l'image de la balise "beacon_image.png".
Chacune de ces interfaces propose une démo pouvant être utilisée par l'utilisateur ou d'autres programmes du projet

- robot :

Ce sous-module permet la définition du robot sous plusieurs formes :

- robot_simule.py : simulation d'un robot,
- robot_mockup.py : mockup du robot réel pour les tests,
- robot2IN013.py : contrôle du robot réel.

- **Test :** contient les tests unitaires de chaque fonction du projet pouvant nécessiter un test dont les tests des adaptateurs, des robots, de l'affichage ainsi que de l'environnement.
(test_adapt.py, test_adapatateur_simule.py, test_adaptRR.py),

- **Les fichiers exécutables :**

- main2d.py permet de démarrer l'interface2D.
- main3d.py permet de lancer l'interface3D.
- mainmock.py permet d'exécuter les deux interfaces (une à la fois) sans fermer le programme (la position des robots reste la même d'une interface à l'autre).
- main_reel.py permet au robot réel d'exécuter les différentes stratégies du contrôleur.
- menu_reel.py est une interface (probablement en console) pour interagir avec le robot réel.

Il est nécessaire de noter que, dans chaque module ou sous-module, il y a un fichier **__init__.py** qui facilite les imports en récupérant les méthodes, constantes ou instances qui pourraient être nécessaires en dehors du module.

Gestion du travail de groupe

Pour la gestion de notre projet, nous avons utilisé deux outils principaux : **GitHub** pour la partie code et **Trello** pour la partie organisationnelle. Ces outils nous ont permis, tout au long du projet, d'avoir une prise en main accompagnée et facilitée.

La fragmentation des objectifs en tâches unitaires nous a permis de mieux nous organiser et de répartir le travail entre les membres du groupe. La répartition se faisait ensuite de manière assez naturelle, chacun prenant en charge des tâches spécifiques.

GitHub a été très utile d'un point de vue pratique, car il nous a permis de travailler à plusieurs sur le projet, tout en maintenant une bonne organisation et synchronisation de nos développements.
Nous organisions des **réunions hebdomadaires** afin de faire le point sur l'avancée des tâches, mais surtout pour partager d'éventuels problèmes rencontrés et réfléchir ensemble à des solutions.

Conclusion :

Ce projet de robotique a permis de concevoir et de mettre en œuvre un système de contrôle avancé pour un robot GoPiGo, en combinant simulations 2D et 3D ainsi que l'utilisation d'un robot réel.

Grâce à l'utilisation de Pygame et Panda3D, nous avons développé des interfaces graphiques efficaces pour tester et valider des stratégies de navigation dans des environnements simulés. L'intégration des modules de contrôle, d'environnement et de robot, ainsi que l'adaptation au matériel réel via la bibliothèque easygopigo3, a démontré la robustesse de notre architecture.

Malgré les défis techniques rencontrés, la collaboration étroite entre les membres du groupe, soutenue par des outils comme GitHub et Trello, a assuré une gestion efficace du projet.

Les résultats obtenus ouvrent la voie à de futures améliorations, notamment en optimisant la reconnaissance d'image et en étendant les capacités du robot à des environnements plus complexes.

Avis sur le projet :

En rétrospective, ce projet a pu être parfois frustrant, mais il nous a offert une expérience enrichissante et des compétences précieuses. Il nous a permis de collaborer efficacement en équipe sur notre premier projet d'envergure, tant par le temps investi que par les objectifs atteints et le travail fourni.

Nous avons été confrontés à divers défis techniques, nous obligeant à chercher des solutions en nous appuyant sur des forums, nos camarades de groupe ou d'autres étudiants.

Ce projet s'est révélé particulièrement captivant grâce à l'application concrète de nos efforts avec un robot physique réagissant à nos commandes, une nouveauté par rapport aux travaux précédents limités à nos ordinateurs. Voir nos programmes prendre vie dans le monde réel et observer les progrès hebdomadaires de notre projet, parti d'un dossier vide, a été une source de satisfaction et de fierté.

Remerciements :

Nous tenons à exprimer notre gratitude envers nos deux professeurs, M. Baskiotis et M. Sigaud, pour leur accompagnement tout au long de ce projet.

Merci pour vos conseils avisés prodigués chaque semaine. Les séances avec vous, bien qu'exigeantes et parfois décourageantes au début, nous ont appris la rigueur, ont renforcé notre motivation et nous ont permis de mieux structurer notre travail.

Votre encadrement, votre soutien constant, ainsi que votre expertise et votre patience ont été essentiels pour surmonter les difficultés et avancer de manière significative.

Nous vous sommes sincèrement reconnaissants pour tout cela. Nous espérons avoir démontré nos compétences et vous avoir convaincu, en tant que client, de faire confiance à notre équipe pour la réalisation de votre projet !