Université Grenoble Alpes

Plan de développement

Projet Intelligence Artificielle

Eva DROSZEWSKI
Sara IKAN
Anesie MARTINIANI
Alix PRATABUY

I- Introduction : Objectif du document

Ce document a pour but de planifier le développement de notre robot Sully dans le cadre d'un projet de robotique pour un cours d'initiation à l'intelligence artificielle. Le but est de ramasser un maximum de palets, dans un temps limité, lors d'une compétition contre d'autres robots.

II- Organisation du projet

1. Outils utilisés

Pour le développement de notre code, nous avons utilisé La Javadoc de LeJOS (java for Legos Mindstorms), notre cahier des charges ainsi que notre échéancier. Pour la programmation, nous avons utilisé l'environnement Eclipse.

2. Étapes principales

Phases	**Livrables**	**Date limite**
Analyses des besoins	● Cahier des charges ● Plan de développement	Semaine 3 Semaine 5
Développement du code	● Tests ● Documentation interne ● Code source	Semaine 10 Semaine 11 Semaine 11
Évaluation	● Compétition ● Rapport final	Semaine 12 Semaine 13

3. Répartition des tâches

Nous avons divisé le groupe en deux équipes constituées respectivement de deux personnes. Une équipe était chargée de s'occuper du développement de la classe Actions et une autre équipe de la Classe Capteurs. Le programme principal a été réalisé ensemble. Le tableau en annexe I récapitule la répartition des tâches selon les équipes.

III- Approche de développement

1. Création d'un Automate à état finis

Nous avons d'abord établi un automate pour rassembler tous les états par lesquels le robot doit passer pour réussir et respecter le règlement de la compétition. Nous avons d'abord créé une première version. Après consultation avec le commanditaire, nous avons apporté des modifications et pris en compte les recommandations formulées (voir Annexe II).

2. Fonctionnalités prévisionnelles

a) Diagramme

Premièrement, nous avons mis sous la forme d'un diagramme les différentes classes et méthodes prévisionnelles selon notre automate.

Description des méthodes par classe

Deuxièmement, nous avons spécifié toutes les méthodes prévisionnelles dans nos trois classes, ce qui nous a permis de répartir efficacement les tâches entre les membres du groupe (voir Annexe I).

Classe Actions

tourner_de(angle) : permet au robot de tourner d'un angle donné (spécifié dans les paramètres de la fonction). Permet aussi de stocker les valeurs de distance pendant que le robot tourne grâce à **stocker_distance.**
ouvrir_pince : permet au robot d'ouvrir ses pinces quand il détecte un palet.
fermer_pince : permet au robot de fermer ses pinces
recherche_palet : permet au robot de rechercher un palet en effectuant un tour complet sur lui-même. Il se dirige ensuite vers la discontinuité détectée et réinitialise la valeur de l'angle.
mouvement_aleatoire : permet au robot de faire un mouvement aléatoire en choisissant une valeur d'angle de rotation et une distance à parcourir au hasard
avancer_sans_palet : permet au robot d'avancer les pinces ouvertes lorsqu'il détecte un palet.
sarreter_sans_palet : Le robot s'arrête lorsqu'il détecte un palet. Il commence à avancer en mesurant la distance. Dans le cas où la distance mesurée passe sous le seuil défini, cela signifie que l'obstacle est un mur, donc il s'arrête pour se réorienter.
attraper_palet : permet au robot d'attraper le palet lorsque la distance est en-dessous du seuil fixé, il ouvre d'abord les pinces, il s'arrête puis ferme les pinces.
tourner_vers_nord : grâce à la boussole, le robot s'oriente vers le nord pour avancer vers la ligne blanche et pouvoir déposer le palet.
avancer_avec_palet : permet au robot d'avancer en ayant le palet entre les pinces (fermées).
eviter_obstacle : lorsque la distance est inférieure à une distance seuil (distance qui correspond à celle où le palet n'est plu dans le champ de détection du capteur ultrason du robot, environ 32 cm) le robot adopte la

stratégie d'évitement qui est de tourner à droite puis tourner à gauche et continuer tout droit (première stratégie envisagée, nous testerons une autre méthode par la suite).
avancer_de(distance) : permet au robot d'avancer d'une distance donnée (spécifiée dans le code).
sarreter_avec_palet : Lorsqu'il a le palet dans les pinces, tant qu'il n'a pas détecté la ligne blanche, grâce à **get_couleur,** mais qu'il détecte un obstacle (avec la fonction **detecter_discontinuite)**, il fait une stratégie d'évitement en utilisant la fonction **eviter_obstacle**. Lorsqu'il l'a franchie, il avance de 10 cm en appelant la fonction **avancer_de(distance)**, puis il ouvre les pinces et recule de 10 cm à l'aide de la fonction **reculer_de(distance)**.
reculer_de(distance) : permet au robot de reculer d'une distance donnée (spécifiée dans le code).

Classe Capteurs

get_distance : permet de récupérer la distance qui le sépare d'un objet à un instant t.
stocker_distance : permet de stocker dans une liste toutes les valeurs de distance à intervalles de temps régulier.
detecter_discontinuite : permet de détecter une discontinuité en comparant la dernière et l'avant dernière valeur de la liste renvoyée par la fonction stocker-distance. Si leur écart est assez important, la fonction renvoie les deux valeurs.
get_pression : permet de récupérer la valeur de la pression à un instant t.
get_couleur : permet de récupérer la valeur de la couleur à l'instant t. Renvoie une chaîne de caractères correspondant à la couleur détectée.
get_position : permet au robot de tourner une fois sur lui-même tout en récupérant les valeurs de distance aux murs et de trouver les deux minimums correspondant aux deux coordonnées de sa position.
get_angle_trigo : permet de récupérer l'angle de rotation du robot. Pour cela nous récupérons la distance au mur avant de commencer à tourner et celle une fois la rotation terminée. Ensuite nous calculons avec une formule de trigonométrie et la fonction nous renvoie l'angle correspondant en degrés.
get_angle_points : permet également de récupérer l'angle de rotation du robot. Dans ce cas, on suppose que l'on connaît le nombre de points de mesure de distance que le robot effectue lorsqu'il fait un tour de 360°. Le calcul de l'angle se fait ici avec un produit en croix, et de même la fonction renvoie l'angle en degré.
choisir_methode_mesure_dangle : permet de dire quelle technique il faudrait utiliser afin de mesurer l'angle. En effet, dans certains cas un objet peut se trouver face au robot avant le début de sa rotation ou à la fin de celle-ci. Dans ce cas la distance mesuré n'est pas celle au mur, et l'angle renvoyé par la méthode **get_angle_trigo** est faux. Pour choisir on compare les deux valeurs renvoyées par les fonctions. Si celles-ci sont trop différentes on privilégiera la méthode **get_angle_points** car plus fiable. Dans ce cas la méthode renvoie l'angle renvoyé par cette méthode et le chiffre 2 pour signaler que la méthode utilisée est **get_angle_points**. Dans le cas contraire, la première étant plus précise, la fonction renvoie l'angle déterminé par la méthode et le chiffre 1 pour signaler que la méthode utilisée est **get_angle_trigo**.
get_orientation : permet de mettre à jour l'orientation du robot en ajoutant l'angle de rotation et renvoie la valeur actualisée.
stocker orientation : permet de stocker dans un tableau toutes les valeurs d'orientation.

Classe Main

boussole : permet de définir le nord, sud, est et ouest en fonction des différentes valeurs d'angle. On fait une fonction qui permet de reset l'angle à 0 et elle sera utilisée dans la fonction NORD de la boussole.
programme général : code principal permettant de traduire les choix effectués par le robot en fonction de la

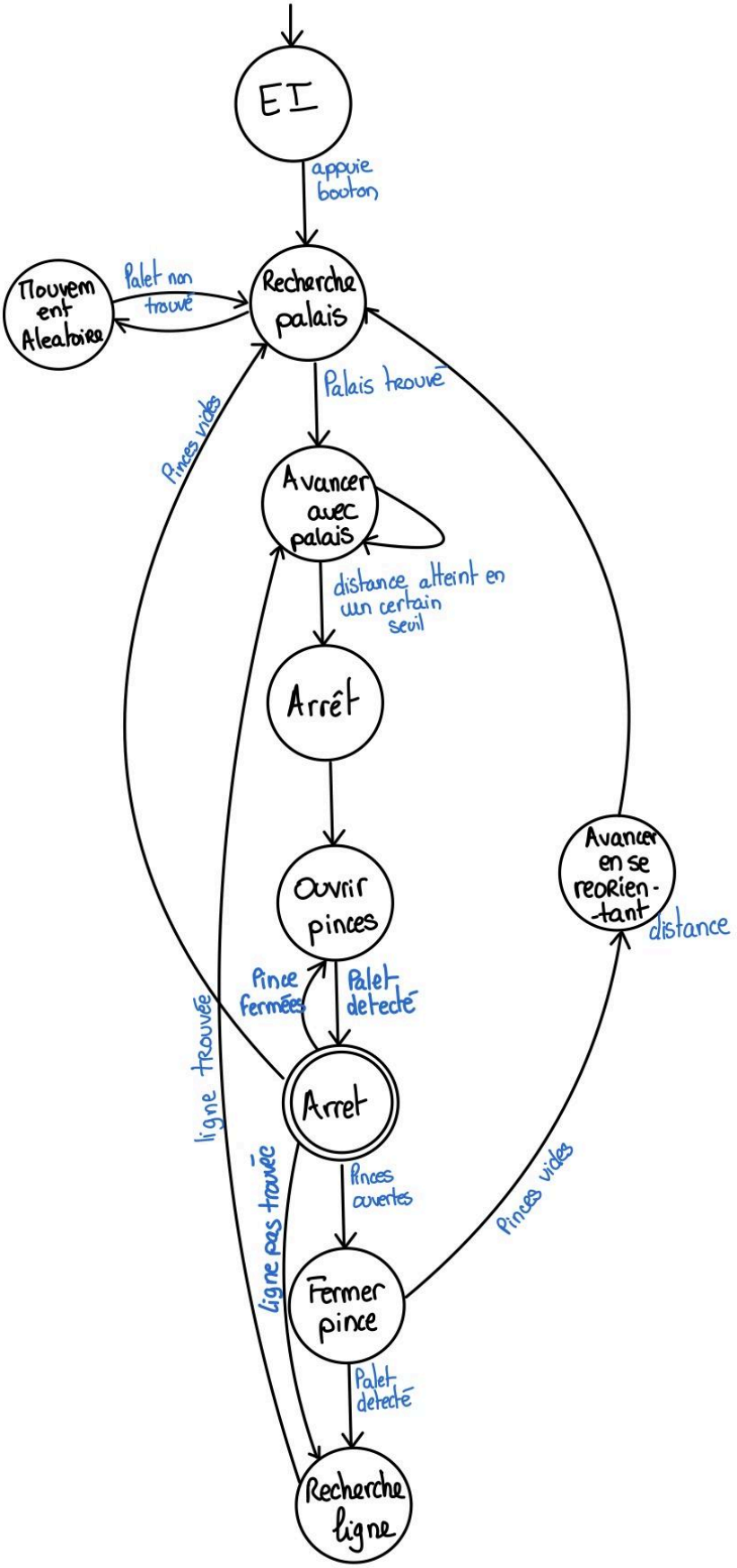
perception de son environnement

IV- Annexes

1. Annexes I : Répartition des tâches au sein de l'équipe

	Semaine 5	Semaine 6	Semaine 7	Intéruption pédagogique	Semaine 8	Semaine 9	Semaine 12	Semaine 13	Semaine 13
Tâches	07/10-13/10	14/10-20/10	21/11-27/11	28/09-03/11	04/11-10/11	11/11-17/11	02/12-08/12	09/12-15/12	16/12-22/12
Classe Actions									
tourner_de(angle)									
ouvrir_pince									
fermer_pince									
recherche_palet									
mouvement_aleatoire									
avancer_sans_palet									
sarreter_sans_palet									
attraper_palet									
tourner_vers_nord									
avancer_avec_palet									
eviter_obstacle									
avancer_de(distance)									
sarreter_avec_palet									
reculer_de(distance)									
tests									
Classe Capteurs									
get_distance									
stocker_distance									
detecter_discontinuite									
get_pression									
get_couleur								Compétition	Rendu du rapport
get_position									
get_angle_trigo									
get_angle_points									
choisir_methode_mesure_dangle									
get_orientation									
stocker_orientation									
test									
Classe main									
boussole									
programme général									
test									
Équipe 1 : Eva et Anesie									
Équipe 2 : Sara et Alix									
Groupe en entier									

2. Annexes II : Automate



3. Annexes III : Diagramme