FLEURY Pierre
JORDAN Célia
JULIARD Victor
KOSAREVA Marguarita

Rapport de projet

Initiation à l'intelligence artificielle

Table des matières

Explication du programme

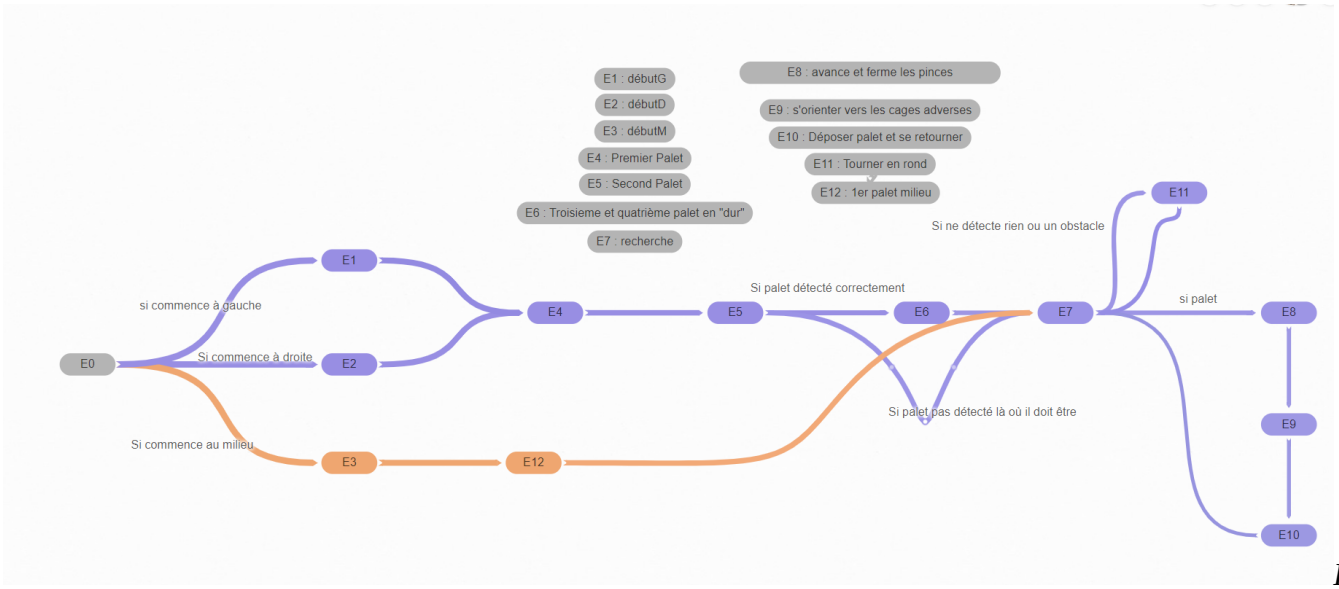

Figure 1: AUTOMATE DE L'AGENT

Cet automate sert à décrire la façon dont notre robot fonctionne sur le terrain et donc notre programme. Il est à noter que si l'on appuie sur le bouton UP à un quelconque moment de l'exécution du programme, le robot s'arrêtera totalement et il faudra alors recommencer toute l'exécution du programme depuis le début.
Nous allons donc décrire brièvement chaque états ce qui fera office d'explication du programme principal.

État initial / E0

Ce n'est pas vraiment un état inclus dans le programme, il décrit la phase durant laquelle nous allons choisir si le robot va partir à droite, au milieu ou à gauche sur la ligne de départ et ainsi lancer des stratégies différentes selon le point de départ.

Début à gauche / E1 et début à droite / E2

Lorsque le robot commence à droite ou à gauche, la même méthode appelée « tactiqueNormale() » mais avec des paramètres dont les valeurs sont différentes est lancée pour permettre d'aller chercher le premier palet en passant le plus à coté du mur une fois le premier palet récupérer (en se décalant sur la droite lorsque le départ se fait à droite, ou à gauche si le départ se fait à gauche).

Cette technique aura permis dans la plupart des cas d'éviter une confrontation avec le robot adverse que nous voulions éviter à tout prix pour remporter les premiers points.

Récupérer et marquer le premier palet / E4

Une fois le choix du point de départ déterminé, la récupération du palet se fera en code dit « dur », aucune flexibilité n'est permise, le robot fonce tout droit pendant quelques dizaines de centimètre sans même prendre en compte si le capteur de poussée avait été activé, ferme les pinces, tourne de 90 degrés à gauche (respectivement à droite selon la position de départ), avance légèrement, se retourne de 90 degrés pour se mettre face au camp adverse puis fonce tout droit pendant quelques dizaines de centimètres puis ouvre les pinces. Le premier palet est ainsi marqué.

Ce choix de n'avoir aucune flexibilité n'est pas forcément la meilleure stratégie mais permet d'assurer la récupération d'un premier palet dans la plupart des cas.

Récupérer et marquer le second palet et le troisième / E5

La récupération de ce second palet s'effectue aussi en code dit « dur », aucune flexibilité permise ici, la stratégie est différente si le robot commence au milieu, mais ce sera expliqué plus bas.

Une fois le premier palet récupéré, le robot recule, se retourne d'un angle exacte de 157 degrés (mesure obtenue lors des phases de test) pour être face au second palet qui est celui qui se trouve derrière le robot, foncer de quelques dizaines de centimètres, refermer les pinces, se retourner de -157 degrés (ou +157 selon la position de départ pour éviter de partir dans le mur, c'est le paramètre « int j » de la méthode tactiqueNormale) puis de foncer derrière la ligne adverse et ouvrir les pinces. Une fois le second palet, l'opération se répète pour le 3ème, même si l'angle pour se retourner n'est pas le même, il est précisé en paramètre de la même manière (« int k » dans la méthode tactiqueNormale)

Encore une fois cette stratégie n'est pas flexible, mais si la récupération du premier palet s'est bien passée il y a peu de chance que le robot adverse soit de notre côté du terrain et nous gêne, cela nous permet ainsi de récupérer un second palet puis le troisième sans trop de difficulté et rapidement.

Récupérer et marquer le quatrième palet si il est bien placé / E6

Lors de la compétition nous avions remarquer que la récupération de palet en code dit « dur » marchait plutôt bien c'est pourquoi nous avons décider de rajouter un quatrième palet à récupérer de la même manière sans pour autant compromettre une éventuelle recherche mais aussi en nous adaptant à la stratégie de l'adversaire durant la compétition, c'est pour cela que quelques lignes de code qui peuvent parfois paraître un peu confuse avant de passer sur la recherche sont présentes dans la version finale du programme.

Une fois le troisième palet récupéré, le robot recule et se retourne une nouvelle fois. Dans le cas où le quatrième palet est placé correctement, le robot va le détecter à l'aide de son capteur ultrason (on regarde si la distance perçue est inférieur à un certain seuil, sachant que le robot est déjà orienté correctement, on sait ainsi si le palet est présent ou non) et se diriger vers lui, une fois la distance parcourue atteinte (mesurée au préalable, elle est fixe), le robot ferme les pinces et se retourne.

A ce moment là, la méthode recherche() est appelée mais les pinces sont fermées donc on sort automatiquement de la méthode, la méthode deposerPalet() est donc appelée, amenant le robot à avancer (il s'est retourné après avoir récupérer le palet il est donc orienté face aux cages adverse) jusqu'à ce que la distance captée soit inférieur à un certain seuil, indiquant qu'il est arrivé contre le mur

du fond, il peut donc déposer le palet et se retourner. Il passera par la suite directement à la méthode recherche

Si le palet n’est pas bien placé : méthode recherche() / E7

Dans le cas où le quatrième palet n’est pas bien placé, (le robot ne détecte pas d’objet à la distance connue) ou alors lorsque celui ci vient d’être ramassé, il passe tout de suite dans l’état « recherche de palet » à l’aide de la méthode recherche. On initialise un attribut appelé ici « angle » qui sera incrémenter (ou décrémenter selon le sens de rotation) à chaque rotation du robot pour lui permettre à chaque fois de se repositionner face aux cages adverse par la suite.
Celle ci est en marche tant que les pinces du robots sont ouvertes, tant que le robot ne détecte rien dans un certain intervalle et que le capteur de poussée n’est pas actionné. Si il ne capte rien dans un intervalle de distances donné, alors il va tourner sur lui même, 20 degrés par 20 degrés (cela incrémente à chaque fois le compteur angle de 20) jusqu’à capter quelque chose.

Le désavantage de cette méthode est le fait que le robot tourne de 20 degrés par 20 degrés, ce qui manque de précision, de plus l’intervalle de distance de détection pour considérer que nous avons affaire à un objet peut amener à ne pas détecter un palet qui pourrait nous avantager plus qu’un autre.

Si un palet est détecté / E8

Si quelque chose rentre en vue du robot alors, il va avancer d’une distance donnée. Si une fois cette distance parcourue la distance détectée est supérieure à un nombre donné, c’est alors que nous avons affaire à un palet (puisque le robot ne le détecte subitement plus, étant donné la hauteur des palets et la vision en cône du robot), on fonce donc vers le palet d’une distance donnée, puis on ferme les pinces, ainsi on sort de la méthode recherche.

Se réorienter et ramener le palet dans les cages adverses / E9 et E10

Une fois sortie de la recherche, une dernière action s’effectue avant de sortir réellement de la méthode : se réorienter, grâce à l’attribut angle, le robot va se retourner de « 180 - angle » degrés pour ainsi faire face aux cages de l’adverse.

La méthode déposerPalet() est donc appelée, et le robot va avancer jusqu’à ce que le capteur de distance détecte un mur, une fois cela effectué, le robot ouvre les pinces et se retourne pour relancer une nouvelle fois la méthode de recherche (on retourne à l’état E7). Ces étapes sont répétées ainsi plusieurs fois afin d’avoir une chance de récupérer tous les palets, le problème c’est qu’il y a de grandes chances que le robot reste bloqué si il ne détecte rien dans l’intervalle de distance donné, de plus le robot adverse peut grandement perturber cette méthode.

Un obstacle est détecté / E11

A chaque rotation de 20 degrés, lors de la recherche, si quelque chose est détecté en dessous d’une certaine distance seuil, alors c’est que nous avons affaire à un mur, on choisit donc de continuer de tourner de 20 degrés et on repasse dans la recherche d’un objet dans un certain intervalle de distances.

Départ au milieu et premier palet si départ au milieu / E3 - E12

Si le départ du robot s'effectue au milieu, n'ayant pas pris le temps d'élaborer une stratégie plus poussée ni de faire des mesures pour coder la récupération d'autres palets en code « dur » le choix a été fait de ne programmer que la récupération du premier palet de la même manière que pour le début à gauche puis par la suite de passer directement dans l'étape de recherche de palets.

C'est une stratégie que nous voulions éviter de mettre en place car moins efficace que les autres, elle n'aura été mise en place que lorsque nous avions besoin de nous adapter à la stratégie de l'adversaire et que les autres méthodes étaient donc gênantes.

Conclusion automate

La partie code « dur » permet de récupérer de nombreux palets sans trop de difficulté (en supposant que le robot adverse ne nous gêne pas trop) ce qui peut s'avérer efficace dans certains cas. Néanmoins une méthode de recherche est mise en place même si elle n'est pas optimale, elle permet au bout d'un certain temps d'avoir tout de même une assez grande flexibilité. Le gros problème que nous rencontrons ici c'est le fait de devoir passer à chaque fois par la récupération des premiers palet avant de passer sur la recherche ce qui peut être un gros frein pour gagner la compétition en cas de relance du robot. Le cas où nous serions face à un autre robot n'a pas été prévu si l'autre n'a pas prévu non plus de nous faire face nous entrons donc en collision ce qui peut être gênant pour le match si il n'y a plus de temps mort disponible.

Idées abandonnées

Les classes « Demo »

Des classes de démonstration contenant des méthodes pré-faite à tester sur le robot pour tester ses fonctionnalités ont été trouvé sur internet et devaient servir de base pour créer le programme principal. Ces classes sont les suivantes : UltraSonicDemo, ColorDemo, TouchSensor, DriveAvoid (reculer et tourner à chaque fois qu'un obstacle est détecté), et DriveSquare (avancer et tourner en carré). Mais chacune de ces classes testaient les fonctionnalités du robot de façon indépendantes et les diverses méthodes présentes dans ces classes étaient trop simplistes pour être utilisées telles quelles. Seules quelques méthodes ont étés retenues et ré-implémentées dans la classe Capteurs notamment pour le capteur de couleur.

Les méthodes de la classe action

Il était initialement prévu d'utiliser quelques méthodes qui ont finalement été retirée du code final :
- void tourner90(char orientation)
- void demiTour()

La méthode tourner90() devait permettre au robot de tourner à un angle de 90 degrés à gauche ou à droite en fonction du charactère 'g' pour gauche ou 'd' pour droite passé en paramètre. La méthode demiTour() quant à elle devait permettre au robot de tourner à 180 degrés. Mais créer plusieurs méthodes pour faire tourner le robot selon un angle voulu aurait été fastidieux et très peu flexible, la méthode « oriente(int angle) » qui permet de tourner selon un angle plus précis et sans pour autant démultiplier les méthodes aura été préférée, c'est donc cette dernière qui sera gardée.

Les méthodes de la classe capteurs

Le calibrage du capteur ayant été le plus remodelé a été celui du capteur de couleurs. En effet plusieurs approches étaient possible quant à l'utilisation de ce capteurs, c'est pourquoi certaines idées n'étant pas assez optimisées ou utiles ont étés abandonnées. Plusieurs de ces méthodes proviennent de la classe ColorDemo citée plus haut, néanmoins elles font partie de celles que l'on aurait pu garder telles quelles.
- Une méthode renvoyant le taux de luminosité nous aurait permis de calibrer le capteur de couleur dans des conditions variable pour éviter les problèmes de perception dû aux changement de lumière, mais par manque d'échantillons et le peu d'utilité étant donnée les conditions de la compétition : l'idée a été abandonnée.
- Une méthode renvoyant le code RGB (un numéro entre 0 et 255 représentant l'intensité de chaque couleur primaire) de la couleur captée aurait pu être utilisée (à noter que la méthode renvoyait des valeurs selon la luminosité perçue et non pas le réel code RGB qui aurait permis de reproduire la couleur) pour avoir une plus grande précision et notamment faire la différence entre le noir et le bleu que notre robot avait du mal à différencier.

Au final les stratégies que nous avons mis en place n'avaient pas besoin du capteur de couleur, son utilisation devient donc nulle mais dans d'autres conditions et d'autre cas de figure il aurait vraiment pu être intéressant de l'utiliser.

Les méthodes de la classe environnement

Il était initialement prévu d'utiliser une classe appelée « environnement » dans laquelle nous avions penser à décrire l'environnement dans lequel le robot devait effectuer ses actions et évoluer.

Elle devait servir au robot pour se repérer mais il existait aussi un attribut palet initialisé à 9 au lancement du programme et il devait être décrémenté à chaque fois qu'un palet était ramassé, cela aurait permit de savoir quand est-ce que le robot devait s'arrêter. Ne prenant pas en compte les palets ramassés par l'adversaire cet attribut n'a au final que peu d'utilité dans le cadre de la compétition.

Une méthode « String getPos() » était censée retourner la position du robot sur le plateau en fonction des deux dernières couleurs détectées, malheureusement cette stratégie aurait été trop coûteuse et fastidieuse à implémenter en raison du trop grand nombre de possibilités pour derrière n'en avoir que très peu d'utilité, puisque la position n'aurait de toute façon pas été précise : le robot se serait situé par rapport aux 18 cases du plateau (16 cases + les 2 espaces derrières les lignes blanches) qui auraient dû être représentées sous forme de carte dans la mémoire du robot en associant à chaque case les couleurs qui devaient être perçues pour considéré que l'on soit dans une case plutôt qu'une autre.
Une autre méthode nommée « perdu() » aurait dû mettre le robot dans l'était « perdu » au cas où il ne saurait plus où il est et par la suite lui permettre de se repositionner mais il a été préférable au final d'utiliser directement la méthode de recherche de palet qui répond mieux à la problématique.

Gestion du projet

Au début du projet, environ deux heures (parfois quatre) chaque lundis étaient réservées afin de travailler en groupe, et effectuer divers tests sur le robot. Chaque semaines des objectifs précis à remplir pour la semaine d'après étaient définis, des réunions pouvaient avoir lieu entre deux lundis si le besoin s'en faisait ressentir par un des membres du groupe.
Il était aussi initialement prévu d'utiliser « Git » comme service de gestion et de développement logiciel mais avec l'arrivée d'un second confinement et le groupe n'étant pas formé et ne maîtrisant que très peu la plateforme à l'arrivée de ce confinement elle a été délaissée au profit de la plateforme « discord » pour communiquer bien que ce ne soit pas une plateforme dont l'utilisation soit orientée vers la gestion de projet informatique, elle aura tout de même facilité la communication et parfois le transfert de fichiers étant déjà maîtrisée des membres de l'équipe.
L'arrivée de ce confinement a grandement bousculer le développement qui était initialement prévu, certains membres étant dans l'impossibilité d'accéder au robot pendant plusieurs semaines en dehors des quelques heures mises en place les deux lundis avant la compétition, combiné avec des difficultés pour maîtriser l'échange de programmes, certains n'ont pas pu s'impliquer et permettre l'avancée du projet autant que prévu. Le plan de code/échéancier fournit dans ce lot de document n'est donc pas représentatif de la façon dont le projet a été mené et n'est qu'une ébauche de la manière dont les choses auraient dû se passer.

Conclusion

Pour conclure, malgré de nombreux changement et quelques difficultés de gestion du travail et de communication, le projet a su aboutir à quelque chose de correcte. Beaucoup de choses seraient à reprendre ou à améliorer mais tout a été fait pour le mieux en prenant en compte les nombreuses difficultés à surmonter qu'elles soient dû à notre manque d'expérience dans la programmation, notre capacité à gérer un projet ou encore le confinement qui nous a beaucoup ralenti. Malgré tout cela, il en résulte de bonne chose, un apprentissage profond sur la manière dont il faudra procéder à l'avenir si ce genre de projet devait de nouveau voir le jour, nous pouvons maintenant être conscient des choses à refaire ou à éviter dans le futur pour arriver à quelque chose de plus aboutit dans le cadre d'un projet informatique collectif.