Phase ingénierie Système :

TOUT CE QUI EST EN ROUGE A ETE MIS SUR LE LATEX (pour pouvoir plus facilement voir les modif faites) donc écrivez en noir svp

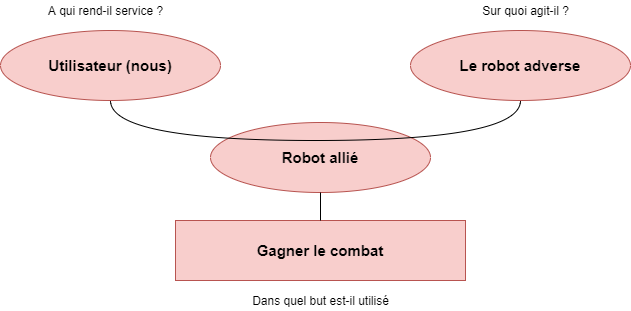
Nous cherchons à construire un robot sumo autonome dont l’objectif est de remporter un combat face à un autre robot sumo. Ce combat se déroule sur un dohyo d’environ 1m². Un robot est déclaré vainqueur lorsqu’il pousse l’autre en dehors du dohyo ou bien, en cas d'égalité à la fin du temps imparti, le robot le plus léger l’emporte.

Pour réaliser ce robot sumo, nous nous appuyons sur le “Bioloid Beginner Kit” ainsi que sur des accessoires en papier ou en carton.

Un tel robot cherche donc à répondre au besoin : “gagner le combat de sumo”.

Le robot devra donc être capable, de manière autonome, de pousser son adversaire en dehors du dohyo. De plus, il doit effectuer cette action sans être endommagé lui-même.

Diagramme de la bête à corne : rend service à nous + agit sur le robot adverse → objectif gagner le combat.



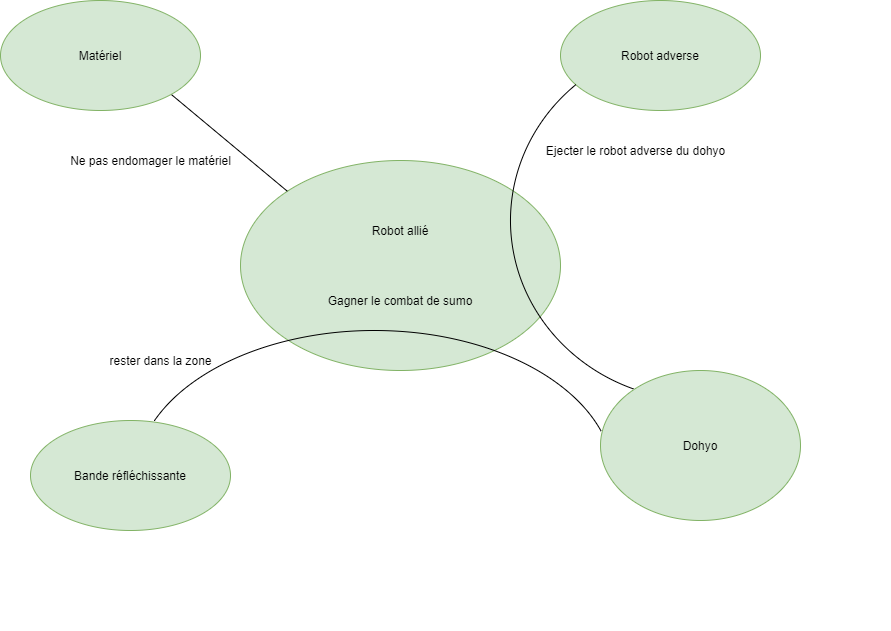
Pour réaliser cet objectif, le robot sumo doit donc avoir comme fonctions principales de se déplacer, d’attaquer son adversaire, de se protéger et de détecter ou repérer des objets.

Mais il doit aussi satisfaire certaines fonctions contraintes notamment rester dans le terrain, commencer à se déplacer que 3 secondes après son allumage et ne pas s’endommager lors du combat.

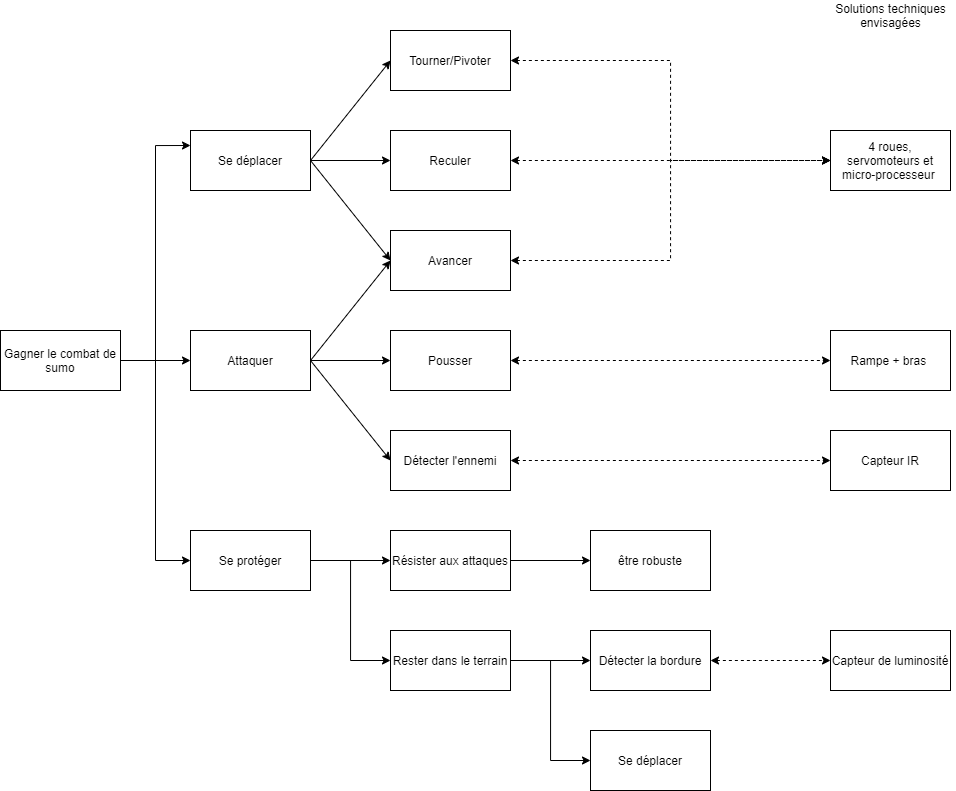
| Fonctions principales | Description | Critères | Précision |
| --- | --- | --- | --- |
| FP 1 | Se déplacer | ?? |  |
| FP 2 | Attaquer | vitesse, poids, force, équilibre... |  |
| FP 3 | Se protéger | résister à une attaque frontale |  |
| FP 4 | Détecter/repérer des objets | distance de mesure, précision de la mesure… |  |

| Fonctions contraintes | Description | Critères | Précision |
| --- | --- | --- | --- |
| FC 1 | Rester dans le dohyo | 1m \* 1m | 20% |
| FC 2 | Mouvement après 3s | temps |  |
| FC 3 | Ne pas s’endommager lors du combat | Matériaux de construction |  |
| FC 4 | Ne pas être trop lourd pour ne pas perdre au poids | Masse pas trop élevée | <2kg |
| FC 5 | Taille du robot doit être plus petite que le terrain | Taille | <1m |

(Peut-être faire un diagramme de la pieuvre à la fin)



On visualise ces fonctions sur le diagramme suivant :



Ca correspond à quoi les exigences d'assurance de résultat? Genre c'est les capacités minimales que doit avoir/réaliser notre robot?

dans ce cas on pourrait mettre :

Le Robot doit au moins pouvoir éjecter hors du dohyo des objets immobiles (pas trop lourds) (donner un poids min, max?) se trouvant sur celui-ci lors des tests.

Le robot ne doit pas sortir si il se trouve seul dans le dohyo.

Le robot doit rechercher sur le terrain un adversaire.

*Description profil/ phase / cycle de vie*

*Faire une flèche du lundi au vendredi avec les différentes étapes comme selon p.22 cours (à faire à la fin quand on aura fini)*

Exigences techniques :

* 4 roues
* bras sur les côtés

Modes de fonctionnement :

* allumage après 3s
* devoir être toujours branché

**Documentation de référence :**

Pour construire notre robot sumo, nous nous sommes appuyés sur le guide d’utilisateur du robot Bioloid produit par l’entreprise, qui explique le fonctionnement de chaque composant et qui donne aussi des exemples de robots à réaliser. De plus, nous nous sommes également basés sur un document de l’ENSTA intitulé “Mise en oeuvre du robot” qui détaille lui aussi mais de manière plus synthétique comment doit fonctionner le robot ainsi que les caractéristiques techniques propres aux composants, notamment aux capteurs et aux moteurs qui nous ont permis d’envisager des tests précis pour vérifier nos hypothèses de programmation et de construction mécanique. Un document présentant une manière détaillée de coder les robots nous a aussi été utile. Voici les liens des documents mentionnés précédemment :

<https://perso.ensta-paris.fr/~toralba/MO103/docs/guide-bioloid.pdf>

<https://perso.ensta-paris.fr/~toralba/MO103/docs/quick-start-bioloid.pdf>

<https://perso.ensta-paris.fr/~toralba/MO103/docs/programmationCM530.pdf>

<https://perso.ensta-paris.fr/~toralba/MO103/docs/MiseEnOeuvreDuRobot.pdf>

**Management du projet : ~compte rendu des réunions ?**

Le premier soir, tout le groupe s’est réuni afin de trouver des idées et tenter de prendre une décision quant aux plans mécaniques à adopter pour le robot. Le plan mécanique devant correspondre à une stratégie de combat, il fallait en même temps développer des idées de programmation en accord avec la géométrie choisie pour le robot. Le lendemain matin, les choix de géométrie et de stratégie à adopter pour le robot étaient faits. Le plan choisi était de faire un robot à 4 roues avec deux bras à l’avant pour tenter de prendre en tenaille l’adversaire (ainsi qu’une légère pente à l’avant visant à déstabiliser l’adversaire en soulevant une de ses roues).

Une fois ce choix effectué, le groupe s’est divisé en deux sous-parties : la première dont l’objectif était de construire au plus vite le robot et la deuxième était chargée de programmer le robot pour qu’il remporte ses combats. Les deux équipes communiquent fréquemment pour faire part de leurs avancements et préparer les tests qu’il va falloir effectuer dès que le robot sera terminé, que ce soit pour obtenir des données nécessaires à la programmation ou des données mécaniques afin d’optimiser ensuite la construction du robot.

Les roues ont été fixées de telle sorte que le robot soit le plus bas possible afin d’avoir une stabilité plus importante. Une première idée avait été de fixer deux roues sur la partie centrale et les deux autres sur chacun des bras mais nous l’avons rejetée car cela aurait empêché le robot d’avancer facilement. Les bras ont ensuite été fixés en assemblant les pièces restantes. Nous avons dû modifier l’extrémité des bras qui initialement étaient constituées de deux grandes plaques. En effet, celles-ci touchaient le sol et auraient engendré des frottements qui pouvaient freiner le robot donc nous avons opté pour des pinces afin que le robot conserve un design original. Ensuite, le capteur a été fixé à l’avant du robot entre les deux bras. Cependant, de rapides tests ont montré que les capteurs saturaient car ils étaient trop proches du sol. Nous avons donc réhaussé le capteur dans l’attente des tests en conditions réelles pour déterminer s’il détecte correctement les bandes réfléchissantes du bord du terrain.

Le mercredi matin, après une journée et demie de travail, la construction mécanique du robot était achevée et les tests pouvaient commencer. Du côté de la programmation, la stratégie à adopter a été affinée comment ? et une simulation du combat a été préparée comment ? afin de programmer un code pouvant s’adapter au mieux aux différentes stratégies que l’adversaire aurait pu choisir.

**Choix des tests/plan de validation/plan de vérification à faire :**

**Premiers tests unitaires réalisés:**

Les différentes fonctions du robot ont été testées séparément dans l’ordre dans lesquelles elles ont été mises en place.

* Tests sur les capteurs: ce sont les premiers à avoir été réalisés par l’équipe programmation car le corps du robot était parallèlement en construction par l’autre équipe. Une fonction affichant les valeurs brutes des capteurs a permis de déterminer les seuils à utiliser pour la détection infrarouge du robot adverse, ainsi que celui utilisé pour la détection de la bande lumineuse. Il a aussi permis d’invalider l’utilisation du capteur de luminosité (résultats parfaitement non exploitables).
* Tests sur un moteur unique: afin de tester les différentes vitesses ainsi que le fonctionnement du code embarqué. Il s’agit simplement de fixer la vitesse.
* Tests sur plusieurs moteurs: afin de vérifier que les moteurs tournent dans le bon sens et les vitesses à utiliser pour tourner.
* Tests du buzzer et des LEDs: vérifier qu’ils fonctionnent, préparer des fonctions permettant de les utiliser facilement et de manière programmable.

Nous avons retenu le capteur infrarouge pour la détection des bandes lumineuses, mais celui-ci avait tendance à saturer lorsque utilisé sur un sol en plastique (comme celui des salles de l’ENSTA non équipé du tapis). Il a donc été désactivé pour les autres tests.

**Mise en place d’une simulation / Programmation plus avancée:**

Nous avons assigné en parallèle du développement un membre de l’équipe à l’élaboration d’une stratégie de combat. Afin de pouvoir tester les stratégies de manière plus simple avant leur implémentation en C et la fin de la construction du robot, il a décidé de mettre en place une simulation numérique en Python du robot et d’un adversaire. Cela nous a permis de nous rendre compte par exemple des stratégies qui ne couvrent pas l’intégralité du terrain.

Le reste de l’équipe de développement s’est consacré à implémenter les fonctions “standard” permettant le contrôle du robot et celles correspondant aux différentes stratégies à adopter en fonction des états.

Une particularité du code retenu est la l’objectif optionnel OO2 nécessitant de jouer de la musique pendant le combat. Ne pouvant interrompre l’exécution du code, nous avons en effet dû mettre en place une horloge en modifiant les fonctions fournies avec le SDK afin de pouvoir jouer des notes à des instants prédéterminés.

**Suite des tests (plus aboutis):**

Une fois la construction du robot globalement achevée, le code arrivait en parallèle à maturité, ce qui nous a permis de procéder à une seconde batterie de tests, cette fois-ci plus longs et en condition réelles (sur le ring).

* Tests de détection d’objets et des bandes réfléchissantes: ces tests nous ont permis de vérifier que le code de détection d’objets et de bandes lumineuses fonctionnait correctement. Il s’agissait dans un premier temps d’un simple retour de valeurs des capteurs, afin de vérifier que les résultats des premiers tests de capteurs s'appliquent bien à la forme de notre robot. Il en est ressorti que nous devions surélever le capteur infrarouge pour éviter la saturation lors de la détection des bandes blanches.
* Tests de déplacements: nous avons vérifié que le robot roulait correctement sur le sol de moquette du dojo, et mesuré sa vitesse pour affiner la simulation. Nous avons aussi vérifié sa capacité à pousser le robot adverse.
* Tests de suivi d’objets et de réaction aux bandes: nous avons vérifié que le robot réagissait correctement en cas de détection d’un adversaire ou d’une bande (qu’il attaquait ou reculait pour rester sur le terrain). Nous avons aussi vérifié (et corrigé), les gestions d’évènements pour que le robot puisse interrompre son action à tout moment si elle n’était plus conforme à la stratégie.
* Tests de la parallélisation de la musique avec les autres tâches: comme expliqué plus tôt, le fait de pouvoir jouer de la musique sans interruption de la stratégie a été difficile, et nous avons vérifié que cette addition ne perturbait pas l’exécution.

**Compte-rendu de la phase de tests :**

* Les tests ont mis en exergue des problèmes de construction mécanique (capteur infrarouge trop près du sol), qui ont nécessité une modification du robot.
* Ils ont permis de mettre au point un simulateur numérique qui nous a aidé pour l’élaboration de la stratégie.
* Ils ont permis de vérifier le bon fonctionnement du code.

(plan de vérification et validation à mettre avant les résultats des tests dans le doc car ils expliquent les tests qui vont être détaillés ensuite)

**Dernier test avant la compétition**

Nous avons constaté que dans la stratégie en L, après lui avoir effectué la forme de L jusqu'au du centre du terrain, nous devions choisir de balayer le terrain dans le sens direct ou le sens horaire en premier.

En faisant des simulations, nous avons déduit que le sens direct en premier nous donnait un avantage sur les autres stratégies.

Cependant, en faisant des tests sur site le jeudi, le robot se mettait à tourner en carré autour de notre point de départ, ce qui nous empêchait de repérer un robot ennemi restant dans sa position de départ.

Pour éviter de tourner en carré, nous avons aussi introduit une asymétrie dans le balayage du robot, pi/2 de balayage dans le sens direct puis 0.83pi dans le sens horaire.

**Plan de vérification :**

Le plan de vérification vise à assurer que le système respecte les exigences du système, il est tout d’abord composé des tests visant à satisfaire les fonctions principales du robot. Il faut faire des tests pour s’assurer que le robot :

* peut bien se déplacer donc effectuer des rotations et des translations;
* peut attaquer et donc être capable de déplacer des objets de dimension similaire à celle que pourrait avoir un robot ennemi;
* peut être capable de résister à une charge ennemi sans s’abîmer ;
* peut détecter des objets et les repérer : en particulier un robot ennemi et la ligne de démarcation du terrain.

Ces exigences fonctionnelles vont souvent de pair avec les fonctions contraintes et donc les tests vont souvent être les mêmes sauf pour vérifier que le robot ne commence à bouger avant 3 secondes.

De plus, comme nous nous sommes fixés des objectifs optionnels, il faut aussi mettre au point des tests pour veiller à leur bonne réalisation :

* vérifier que la musique et les lumières n’empêchent pas le bon fonctionnement du robot (test avec et sans musique pour vérifier qu’il effectue les mêmes actions)
* s’assurer que les lumières s’activent bien en rythme

**Plan de validation :**

Afin de vérifier que notre système réponde bien au besoin spécifié : gagner le combat, nous devons nous assurer que le robot soit autonome lors du combat : libéré dans le dohyo, il doit être capable d’y rester tout en cherchant son adversaire et le cas échéant le pousser vers l’extérieur du terrain.

Nous devons notamment vérifier que notre robot puisse réagir face à différentes options de stratégies de combat de l’ennemi, par exemple lors des simulations informatiques.

**Conclusion :**

*Comment votre démarche de conception s’inscrit-elle dans une démarche d'ingénierie Système ? Quelles approches ont-t-elles et les utilise pour la conception de l’architecture ?*

Suggestion :

Notre démarche de conception s'inscrit dans une démarche d'ingénierie système car nous avons commencé par analyser le besoin rencontré, en prenant en compte l'environnement dans lequel il se trouve. Puis nous avons détaillé les contraintes et les exigences auxquelles notre robot devait satisfaire.

Nous avons ensuite identifié les tâches et les fonctions que notre robot devait réaliser afin de commencer à suggérer des propositions de solutions. Pour ce faire, nous avons divisé le groupe en plusieurs parties, chacune assignée à une tâche bien précise afin de répondre à des exigences bien spécifiques du robot.

Pour la conception de l'architecture nous avons commencé à y réfléchir chacun de notre côté après avoir posé un cadre rapide (le matériel avait été récupéré à ce moment-là).

Ensuite, nous avons exposé nos suggestions, puis débattu chaque idée. L'architecture n'était pas décrite en détail mais nous avions déjà choisi une configuration globale pour notre robot (2 bras à l’avant, 4 roues et nous avions tout d’abord envisagé une rampe à l'avant qui fut abandonnée).

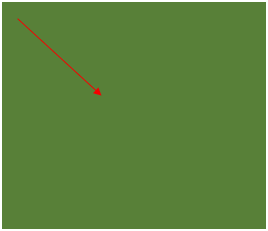
De plus, pour évaluer au mieux les performances de notre robot qui restait simplement théorique dans un premier temps, nous avons fait des simulations avant de réaliser nos tests qui nous ont permis d'améliorer notre modèle.

Les résultats donnés par les simulations ont ainsi pu être comparés aux tests réels, ce qui a permis d’affiner petit à petit les simulations pour se rapprocher autant que possible de la réalité et aboutir à la stratégie la plus efficace possible en combat.

**DÉCISION DE LA STRATÉGIE**

**Phase de recherche des différentes stratégies à adopter au début du combat**

Le robot étant en totale autonomie sur le dohyo, il fallait lui élaborer une stratégie de combat qui soit à la fois efficace contre un maximum d’adversaire et en adéquation avec les paramètres mécaniques et techniques du robot.

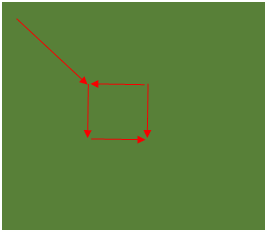
Sachant que le robot sumo va pouvoir adapter sa stratégie uniquement lorsqu’il aura détecté à l’aide de ses capteurs le robot adverse, il faut trouver une stratégie qui permet de repérer l’adversaire dans les meilleures conditions. En effet, une fois que le robot ennemi a été détecté, le robot sumo doit entrer dans une autre phase du programme pour qu’il se prépare au combat. 

* Stratégie naïve

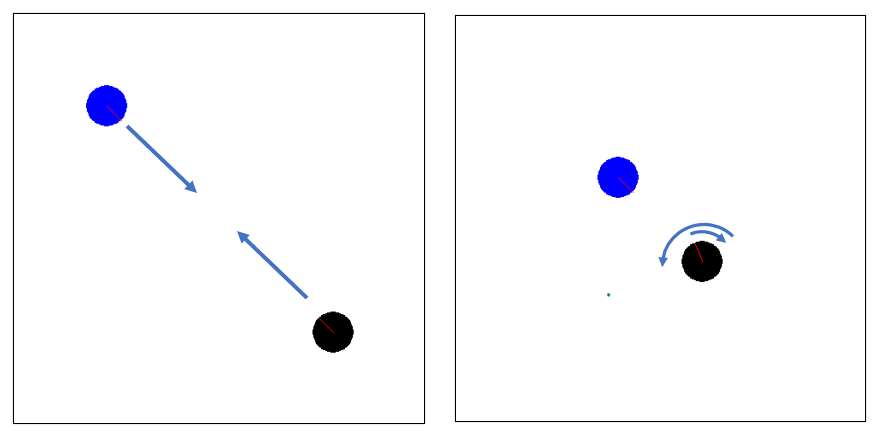
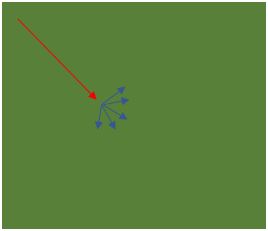
La première stratégie que nous avons ébauchée était simplement de se diriger vers le centre du carré en espérant que l’adversaire adopte une stratégie similaire et que l’on pourrait alors le détecter vers le centre du dohyo.

Cependant, une telle stratégie se doit d’être complétée au cas où l’adversaire ne se trouve pas au centre du dohyo. De ce fait, nous avons imaginé deux poursuites de cette stratégie :

* Stratégie carré
* la première était de parcourir un carré autour du centre du repère en espérant découvrir le robot ennemi lors de ce parcours ;

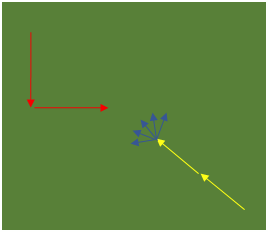
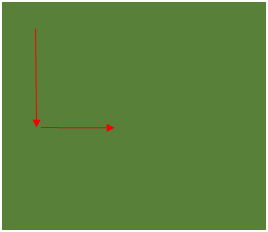


* Stratégie de balayage
* la seconde était de s’arrêter de faire de petites rotations du robot de telle sorte à balayer un cône devant le robot en supposant que le robot adverse ne puisse pas se trouver alors derrière nous ;

(sinon)

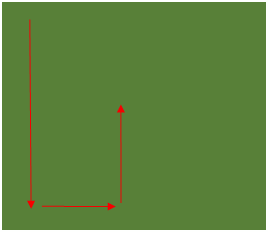
* Stratégie dite en L

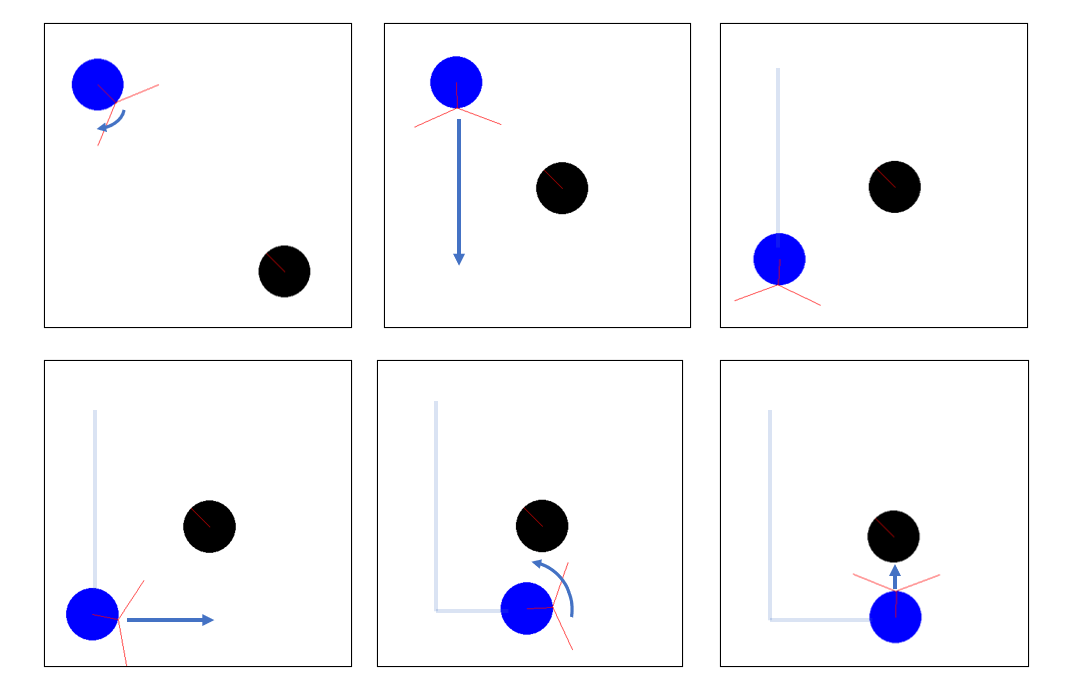
Cependant, il existe d’autres stratégies permettant d’aller vers le centre du terrain. Une d’entre elles serait non pas de suivre la diagonale mais de suivre un bord puis d’effectuer une rotation d’un angle de 90° puis d’avancer à nouveau tout droit afin d’atteindre le centre.

Cette stratégie de déplacement pourrait contrer la première que nous avions embauché. Ainsi, pour tenter de contrer la situation où l’adversaire se déplacerait selon “un L” et nous en diagonale, nous avons pensé à nous déplacer en diagonale mais plus lentement afin que l’adversaire se trouve déjà au centre du terrain lorsque nous arrivons et pouvoir ensuite enchainer sur la phase d’assaut. 

* Stratégie dite en J

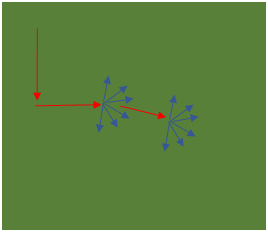
Une autre stratégie envisageable serait de longer un des bords du carré, d’effectuer une rotation de 90°, de revenir vers l’intérieur du carré, toujours en longeant le bord puis de faire une dernière rotation à 90° et enfin de revenir vers le centre pour tenter de prendre l’adversaire par derrière. Nous appellerons plus tard cette stratégie la stratégie “en J”.





En effet, la géométrie de notre robot sumo fait qu’il est très vulnérable s’il se fait prendre par surprise et il faut donc à tout prix tenter d’éviter une telle situation.

Après avoir envisagé toutes ces options, nous avons trouvé qu’une approche “en L” était celle qui était la plus adaptée à notre robot car les bras dont dispose celui-ci lui offraient un meilleur angle d’attaque. Ainsi, il nous fallait encore déterminer notre stratégie de recherche entre le balayage sur place ou le fait de parcourir un carré. Là encore les simulations nous ont montré que parcourir un carré autour du centre du terrain rendait notre robot trop vulnérable car on perdait alors notre avantage que sont les bras à l’avant en exposant les flancs et l’arrière de notre robot. Nous avons donc opté pour une stratégie de recherche avec un balayage puis une phase où l’on avance si jamais nous ne trouvons pas notre adversaire.



Cette stratégie ayant été établie, il nous fallait encore jouer sur certains paramètres afin d’avoir une chance de victoire plus élevée dans tous les scénarios possibles. Les paramètres sur lesquels nous pouvions jouer étaient : la vitesse des différentes rotations du robot, la longueur parcourue sur chaque “branche du L”, l’angle de rotation lors de la phase de recherche et la distance à parcourir après une de recherche infructueuse.

Pour optimiser tous ces paramètres, nous nous sommes appuyés sur la simulation de combat en tentant d’optimiser les paramètres afin de se trouver dans les meilleures conditions pour combattre. Tout d’abord il s’est avéré qu’un angle de balayage total de 180° en utilisant notre vitesse de rotation maximale était la meilleure pour ne pas se faire surprendre et détecter l’ennemi dans un maximum de situations. De plus, elles nous ont permis d’optimiser la durée lors de laquelle il fallait faire tourner les moteurs lors des phases de translation. Nous avons donc choisi une durée de 2640 ms pour la première translation puis une durée de 2350 ms pour la deuxième. Enfin, les simulations nous ont fourni le tableau suivant :

| Stratégies | Tout droit | Tout droit lent | L | L inversé (effectué dans l’autre sens) | J | J inversé |
| --- | --- | --- | --- | --- | --- | --- |
| L (balayage gauche puis droite) | Victoire | Victoire | Victoire | Victoire | Victoire (mieux) | Défaite |
| L (balayage droite puis gauche) | Victoire | Victoire | Victoire (mieux) | Victoire | Victoire | Nul : dépend de l’adversaire |

(si possible sur latex mettre une diagonale avec celle en haut à droite nommée stratégie adverse et en bas à gauche stratégie alliée) -- si c’est possible j’ai regardé comment faire

Ce tableau était censé nous informer de quelle stratégie était optimale entre faire une recherche dont le balayage commençait par la gauche puis vers la droite ou dont le balayage commençait vers la droite puis vers la gauche. Cependant, on voit que ces deux stratégies semblent relativement équivalentes à l’exception de la stratégie adverse qui serait celle du J inversé.

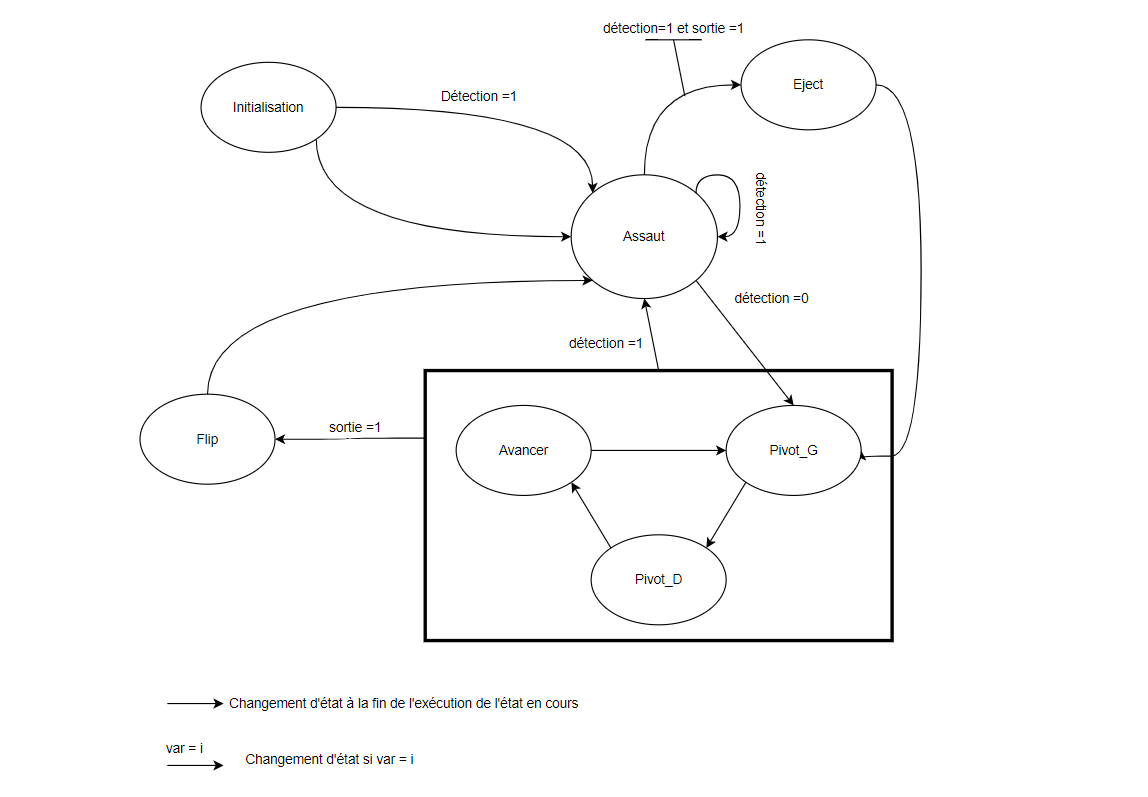
Cependant, nous avons également remarqué grâce aux simulations que si notre balayage commençait vers la droite, alors notre robot n’allait pas observer le coin opposé du dohyo dans le temps imparti. Ainsi, si le robot adverse ne bouge pas du tout, nous n’aurions pas été mesure de le trouver et donc nous nous serions retrouvé dans une situation d’égalité. Or, nous souhaitons éviter au maximum ce cas de figure car nous n’avons pas construit un robot léger pour faire face à cette éventualité. Donc, pour répondre au mieux à notre objectif, il faudrait absolument être en mesure de vérifier ce qu’il se trouve dans le coin opposé à celui de notre départ si l’on ne trouve pas le robot adverse.

**C’est pourquoi nous avons donc opté pour la stratégie d’un balayage vers la gauche puis vers droite pour notre phase de recherche.**

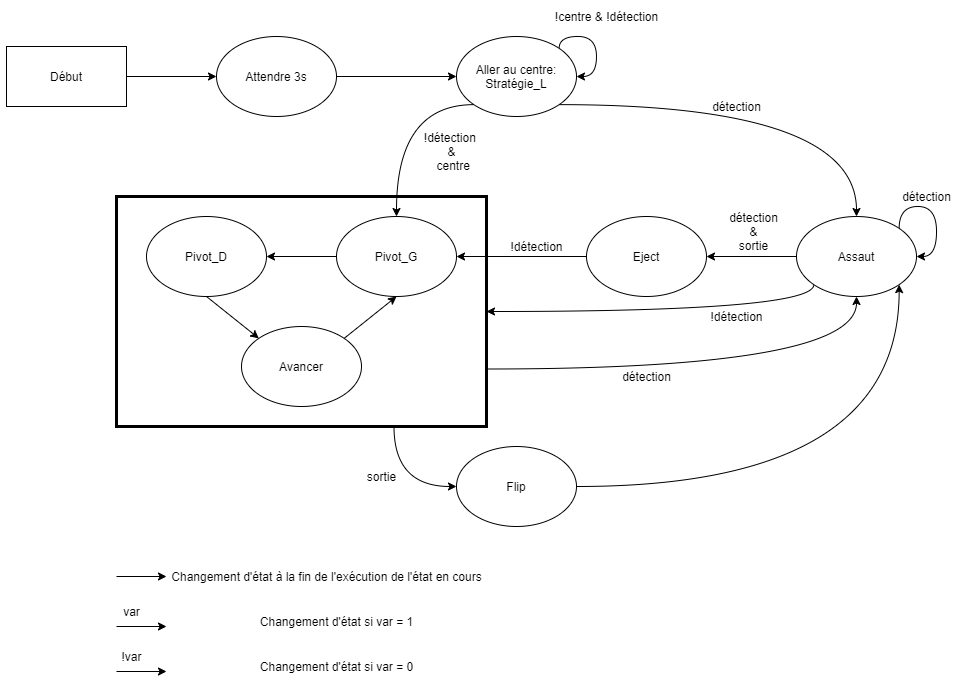
**DESCRIPTION DE NOTRE STRATEGIE**

Finalement, nous avons pu élaborer grâce à nos simulations et à nos tests, l’algorithme final.

son principe est détaillé figure x.



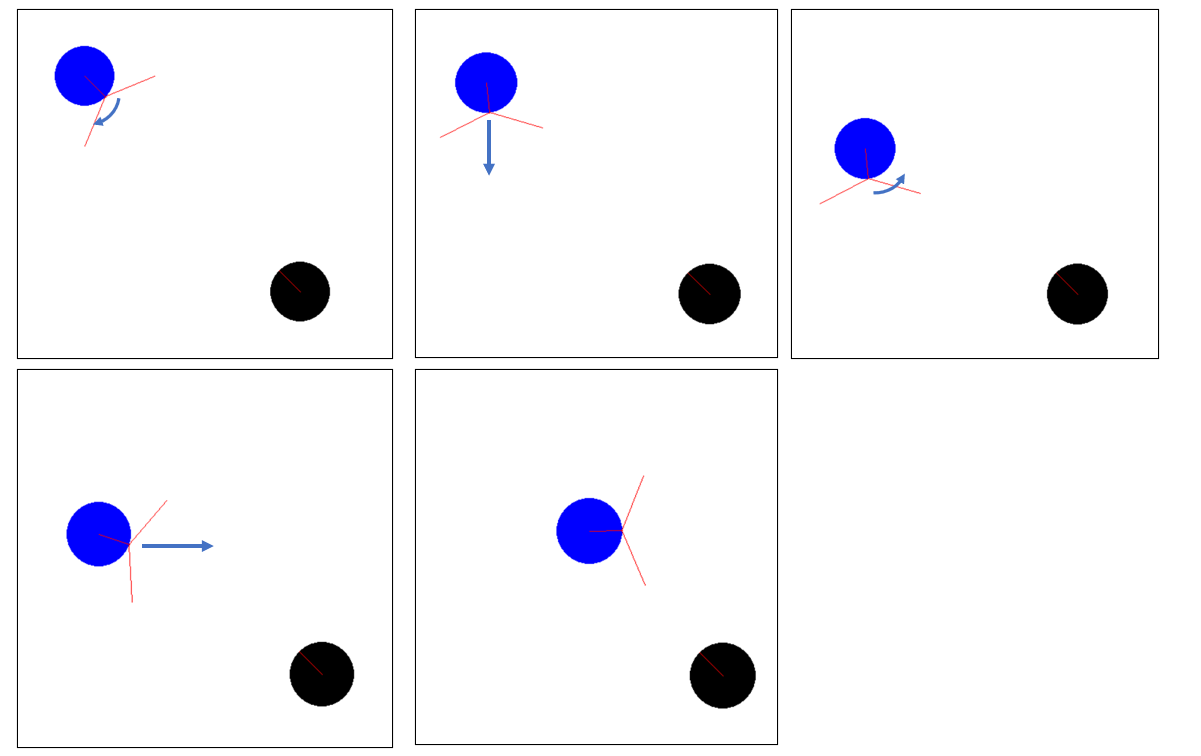
Puis en détaillant la phase d’initialisation, qui comprend le temps d’attente de 3s exigé par le cahier des charges et la stratégie initiale “en L” adoptée on obtient la (figure x+1)



Attendre 3s : Le robot ne démarrera pas avant 3s. La musique démarre.

Aller au centre, stratégie en L : Le robot positionné face à la diagonale centrale, se tourne colinéairement au bord du dohyo choisi. Puis se déplace jusqu’au centre du bord, puis effectue à nouveau une rotation vers le centre puis avance jusqu’au centre (figure x+2).

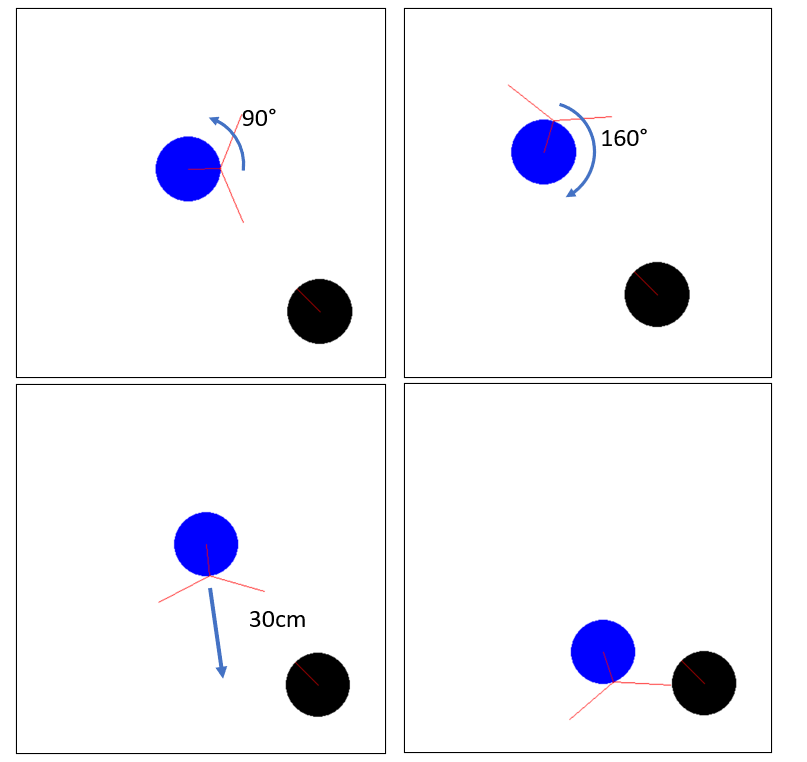
figure x+2 : captures de la simulation de la phase de la stratégie en L (finale) sans détection:



Phase de recherche ( pivot\_D; pivot\_G;avancer ):

* pivot\_G : Le robot effectue une rotation de 90° dans le sens trigonométrique;
* pivot\_D : Le robot effectue une rotation de 160° dans le sens horaire;
* Avancer : Le robot avance en ligne droite sur 30cm.

figure x+3 phase de recherche :



Assaut : On se dirige vers le robot adverse tant que celui-ci est détecté. La vitesse à laquelle le robot se déplace est également augmentée.

Eject : On avance/pousse le robot adverse d’une certaine distance afin de s'assurer que notre robot reste dans le dohyo. Puis on recule, on fait demi-tour dans le sens horaire et on effectue la phase de recherche.

Flip: On recule, on fait demi-tour dans le sens direct et on entre en phase recherche.

Tableau des vitesses lors de la translation et lors de la rotation

| Vitesse des roues lors de : | Initialisation | Recherche | Eject | Assaut | Flip |
| --- | --- | --- | --- | --- | --- |
| la translation (cm/s) | 9 | 16 | 9 | 16 | 9 |
| la rotation (rad/s) | 1,14 | 1,94 | 1,14 | / | 1,14 |

// ajoute le fait qu’on ait dû changer la taille de nos robots dans la simulations pour trop grande différence avec les robots réels

-- points faible pas pris en compte que l’autre robot pouvait être beaucoup plus gros par exemple

**AMÉLIORATIONS POSSIBLES**

Un des problèmes que nous avons rencontré lors des différents tests avec la simulation est que la taille initiale que nous avions envisagée était nettement inférieure à sa taille réelle. De plus, de premier abord, nous avions schématiquement représenté les robots par des cercles mais cela s’est vite révélé être une limite de notre simulation car on ne prenait alors pas en considération l’avantage que représentaient les bras de notre robot. Il a donc fallu adapter la représentation du robot dans la simulation pour essayer de mieux visualiser la stratégie de combat.

Par ailleurs, a posteriori, nous n’avions pas pris en compte la forme que pourrait avoir le robot ennemi. En effet, les robots ennemis avaient souvent une taille bien plus imposante que ce que nous avions prévu. Cette faiblesse de l’étude nous a d’ailleurs coûté cher lors des combats réels.

**DESCRIPTION DE LA SIMULATION**

**Motivation de la simulation**

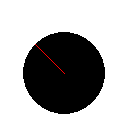
Du fait de la taille importante du groupe et de sa division en sous-section mécanique et programmation, la nécessité d'une simulation visuelle s'est faite ressentir pour permettre à tous d'avoir un support intuitif pour élaborer des stratégies de recherche.

De plus, il devenait possible de confronter différentes stratégies de recherche (foncer tout droit, décrire un L, décrire un J,...) et d'évaluer leurs forces et faiblesses les unes face aux autres, sans avoir à manipuler le robot directement.

**Spécifications de la simulation**

L'objectif de la simulation était d'avoir un outil où les concepteurs de stratégie ont une partie limitée du code à modifier.

-les robots sont représentés par un cercle dont un des rayons est tracé en rouge pour indiquer le placement du capteur et de la face avant du robot;



-les robots peuvent se déplacer en translation dans la direction du capteur ou en rotation;

-le capteur détecte l'adversaire à 20 cm devant lui et s'il est en dehors du terrain et renvoie une valeur binaire;

-les robots ont accès à deux vitesses de translation et deux vitesses de rotation;

-le terrain est représenté à l'échelle par l'écran (480 pixels pour 100cm);

-la simulation s'arrête quand les robots entrent en contact ou quand un robot sort du terrain;

-diviser le code en une partie simulation et affichage et une partie stratégie;

-à chaque pas de temps, les robots ont connaissance de leur vitesse de translation de leur vitesse de rotation, des informations de son capteur, et de variables propres à la stratégie (mais pas celles propres à la simulation comme la position absolue du robot sur l'écran);

-à chaque pas de temps les robots ne peuvent modifier que leur vitesse de translation, leur vitesse de rotation et les informations propres à leur stratégie.

-Plusieurs paramètres sont modifiables:

-Rayon des robots;

-Rayon des robots;

-les 2 vitesses de translation;

-les 2 vitesses de rotation;

-le pas de temps de la simulation.

**Choix des spécifications de la simulation**

-La représentation du robot par un cercle a été choisie pour rendre plus aisé la gestion des collisions dans la simulation et car elle ne semblait pas diverger de la forme réelle du

robot du point de vue du capteur infrarouge.

-La portée de 20 cm du capteur a été prise d'après les tests effectués sur la fiabilité de du capteur infrarouge de l'AX-S1.

-La réponse de l'AX-S1 est binaire dans la simulation car nous l'utilisions de la même manière sur le robot, à l'aide d'une valeur de seuil (le capteur n'était pas considéré comme

assez fiable pour être utilisé plus finement).

-Les deux vitesses de déplacements ont été choisies vendredi matin lorsque nous avons trouvé un moyen de faire tourner les moteurs à une vitesse supérieure à la vitesse maximale

indiquée sur la fiche technique.

-L'arrêt de la simulation lors du contact des robots (c'est-à-dire sa limitation à la simulation de la recherche) est dû au manque de tests physiques sur le frottement des roues sur le dohyo et la méconnaissance de la géométrie du robot adverse. Ces facteurs inconnus nous ont empêchés de prédire la position du robot après le premier contact, et donc auraient nécessité une seconde

simulation plus complexes.

- Un code séparé en une partie simulation et en une partie stratégie évitait aux membres du groupe de programmation l'apprentissage de la simulation dans son ensemble et leur permettait de se concentrer sur l'élaboration de stratégie.

-La position de départ du robot a été estimée à 19 cm du coin du dohyo à l'aide de l'image de dohyo vert du fichier cahierDesCharges.pdf et n'était donc pas considérée comme un paramètre.

-Python a été choisi car connu de toute l'équipe et car le concepteur de la simulation maîtrisait la bibliothèque pygame pour réaliser l'interface graphique. De plus sa souplesse par rapport au C était appréciée.

**Réalisation pratique**

(les noms de variables sont indicés par 1 ou 2 dans le code selon le robot associé)

-Les deux robots sont repérés par leur coordonnées cartésiennes et par un angle qui indiquait la position du capteur du robot (stockés dans posX, posY et orientation).

-Les deux robots ont une vitesse de translation et une vitesse de rotation (stockées dans des variables vit\_trans et vit\_rot).

-Les robots se touchent quand la distance entre leurs centres d<=R1+R2, où Ri sont les rayons des cercles représentatifs.

-À chaque pas de temps, les robots avancent dans la direction de leur capteur de la valeur de leur vitesse de translation, et leurs orientations tournent de la valeur de leur vitesse de rotation.

-À chaque pas de temps, si l'extrémité du rayon rouge indiquant la position du capteur est en dehors du terrain la variable sortie passe à 1 (0 sinon).

-À chaque pas de temps, si les robots sont à distance de capteur, détection passe à 1 si la demi-droite définie par le capteur intersecte le cercle du robot adverse.

-À chaque pas de temps, les fonctions strategie\_1 et strategie\_2 sont appelées. strategie\_i modifie la valeur de vit\_trans\_i, de vit\_rot\_i et de la liste save\_i qui contient les variables propres à leur stratégie que l'élaborateur à décider de sauvegarder. Les arguments d'entrée de ces fonctions sont : vitesse\_trans\_i, vitesse\_rot\_i, sortie\_i, detection\_i et save\_i.

**Précisions sur les variables propres à une stratégie.**

On appelle variable propre à une stratégie\_i à un instant t:

-les valeurs prise par vitesse\_trans\_i aux instants précédents t;

-les valeurs prises par vitesse\_rot\_i aux instants précédents t;

-les résultats de fonction qui n'ont pour arguments que les deux tirets précédents;

Les élaborateurs peuvent conserver une information propre dans la liste save\_i pour la réutiliser dans les itérations qui suivent.

Cette façon alambiquée de conserver des variables simulent la programmation du robot : quand on programme le robot en C, nous n'avons accès qu'au variables des différents moteurs et du capteur et les variables que l'on définit à partir de ces dernières (ce qui est appelé variables propres à une stratégie dans la simulation).

**Précision sur l'intersection entre la demi-droite issue du capteur et le cercle du robot adverse.**

On rappelle que l'intersection entre la demi-droite définie par le capteur et le cercle du robot adverse est censée représenter la détection du capteur infrarouge d'un objet.

Ainsi on ne cherchera à déterminer s'il y a une intersection que si les deux cercles sont à portée de capteur l'un de l'autre.

On ajoute aussi que cette fonction de détection d'intersection fait partie de la partie simulation et non la partie stratégie. De ce fait on s'autorise l'accès à toutes les variables du programme.

On commence par chercher à savoir si le la droite définie par le capteur intersecte le cercle du robot adverse.

Grâce à la position du centre du robot et l'angle de son rayon rouge (vaut 0 dans la position de l'angle nul dans le cercle trigonométrique), on connaît l'équation de la droite définie par

le capteur.

De plus, on connaît aussi l'équation du cercle du robot adverse : on a son centre et son rayon.

Il faut donc prouver l'existence de (x,y) tel que

a'y+b'x+c'=0

(y-y\_2)²+(x-x\_2)²=R\_2²

en réécrivant le système de manière équivalente

a(y-y\_2)+b(x-x\_2)+c=0

(y-y\_2)²+(x-x\_2)²=R\_2²

l'existence d'une intersection devient équivalente à

a²(R\_2²(a²+b²)-c²)>=0

b²(R\_2²(a²+b²)-c²)>=0

qui est équivalent à

R\_2²(a²+b²)-c²>=0

Après avoir vérifié que le robot adverse est croisé par la droite du capteur, on vérifie qu’il est du bon côté (en effet la droite du capteur s’étend à l’arrière du robot).

Pour cela on considère le centre de notre robot comme l’origine d’un repère orthonormé.

Si le robot adversaire est détecté à l’avant du robot, alors son centre est dans le même quadrant que l’avant du robot.

Cela se traduit par

(x\_2-x\_1)\*cos(theta)>=0 et (y\_2-y\_1)\*sin(theta)>=0