Elodie DECERLE
Timothée ISNARD
Younes LAMGHARI
Emmanuelle LERANDY
2ème Année ILE
Semestre de Printemps 2016

RAPPORT
INTELLIGENCE ARTIFICIELLE
PROJET YETI

Encadrants : David JANISZEK &
Marc METIVIER

Sommaire :

I. Tactique

Sachant que le plateau dispose de neuf palets placés aux croisements de chaque ligne. Le robot doit marquer des points en amenant les palets dans le camp adverse rapidement tout en évitant les obstacles (murs c'est-à-dire bordures du terrain, adversaire).

Notre robot dispose d'un système GPS qui lui permet de savoir où il se situe exactement sur le plateau. Il doit notamment être capable de retrouver son chemin à partir de n'importe position sur le plateau. Il dispose aussi des stratégies permettant de réagir aux imprévus.

I. 1- Exécution du programme

Les différentes étapes d'exécution du programme définissent un ordre que le robot devra suivre pour récupérer les palets.

Avant de lancer notre robot, on le place où on veut et puis on choisit les deux couleurs correspondantes à l'emplacement du premier palet qui se trouve devant lui (voir annexe Image 1).

I. **Premier palet :** Le robot est placé au début de la ligne choisie. Il avance tout droit à vitesse moyenne. Quand il détecte le palet, il ferme les pinces puis se met en diagonale et avance au milieu de la case (carrée), se remet en position puis avance tout droit jusqu'à ce qu'il rencontre la ligne blanche. Il ouvre ses pinces (dépose le palet) et recule laissant ainsi le palet dans sa zone de but. Dans cette étape, le suiveur de ligne n'a pas été utilisé pour gagner du temps.

II. **Deuxième palet :** ensuite le robot se tourne vers la ligne (bleu ou verte), il cherche le palet le plus proche (sans suivre de ligne), il détecte le palet, ferme ses pinces puis il fait demi-tour et replace le palet comme précédemment. Notre robot connaît parfaitement les emplacements des palets sur le plateau de jeu. S'il ne détecte pas de palet à son emplacement, il s'arrête puis il va chercher le palet le plus proche encore une fois et ainsi de suite ...

III. **Troisième palet :** Même stratégie que le deuxième palet, on dépose le palet, recule, puis cherche le palet le plus proche encore une fois.

IV. **Les autres palets :** Le robot va refaire la même tactique pour tous les palets jusqu'à ce qu'il n'y ait plus de palets sur le plateau.

II. 2- Système GPS

Le système "GPS" utilise les tachymètres intégrés dans les roues du robot pour mesurer précisément la distance parcourue par chaque roue, et donc en déduire la position et la rotation approximative du robot. Cette position est mise à jour dès qu'une ligne (de position connue) est rencontrée.

III 3- Détecter et esquiver les obstacles

Pour esquiver les obstacles on a deux tactiques différentes :

➔ Si la distance entre notre robot et l'adversaire est inférieure ou égale à 10cm :
On détecte l'adversaire, on fait un demi-tour puis on cherche la ligne noire, pour ensuite la suivre pour récupérer les palets.

➔ Si la distance entre notre robot et l'adversaire est entre 10 et 30 cm :
On détecte l'adversaire, on tourne à droite, puis on avance pour le dépasser et après on tourne à gauche pour revenir sur la ligne de départ.

II. Architecture du code

1. Description des packages

Notre code se compose de six packages :

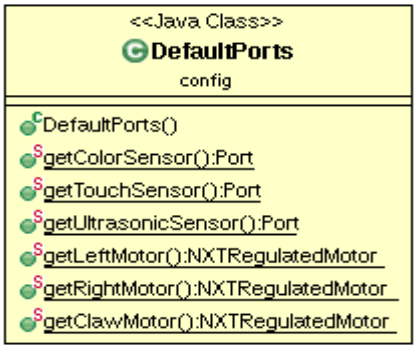
- Package config : constitué d'une seule class qui permet de configurer les ports et moteurs du robot.
- Package gps: constitué de 5 class qui permettent de faire fonctionner le système GPS décrit dans la première partie.
- Package main : avec la class du main où on peut compiler notre programme.
- Package motor : avec deux class qui gèrent les méthodes de mouvements des pinces et des roues.
- Package sensors : avec 6 class qui permettent de faire fonctionner les capteurs (de couleur, d'ultrason)
- Package Strategy : constitué de 12 class qui permettent de gérer les stratégies utiliser par notre robot.

2. Diagramme UML

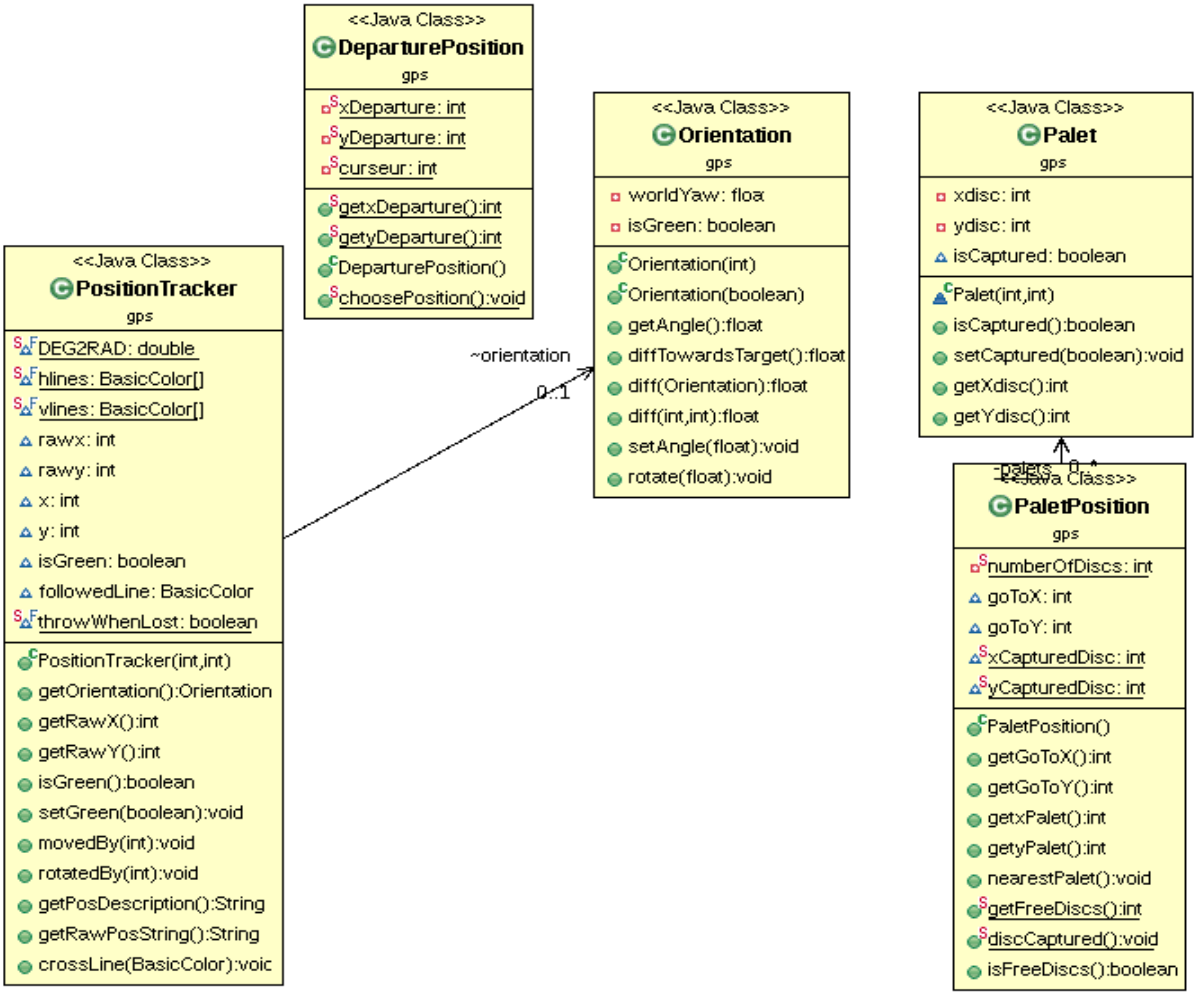
Des diagrammes UML sont fournis pour chaque package Java en format image PNG et au format UCLS. Ils se trouvent à la racine du dossier rendu en haute qualité.

Voici un aperçu de ces derniers :

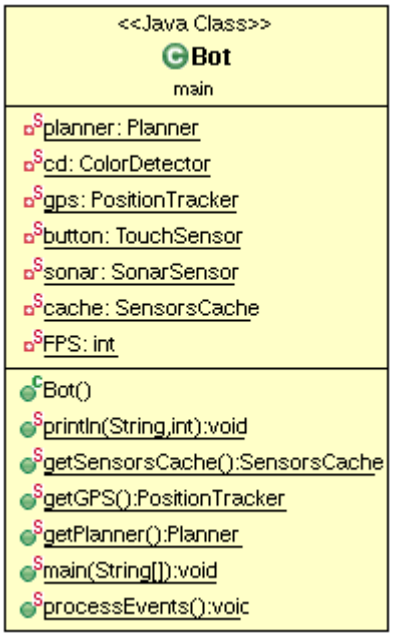
Package config:



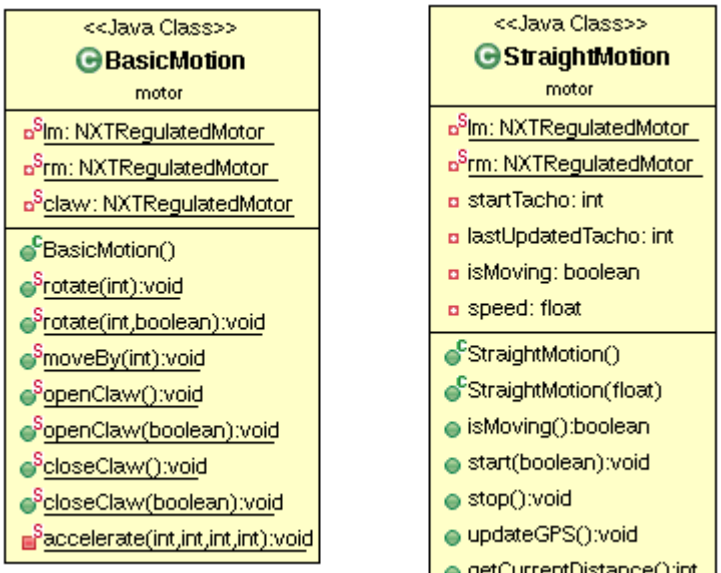
Package gps:



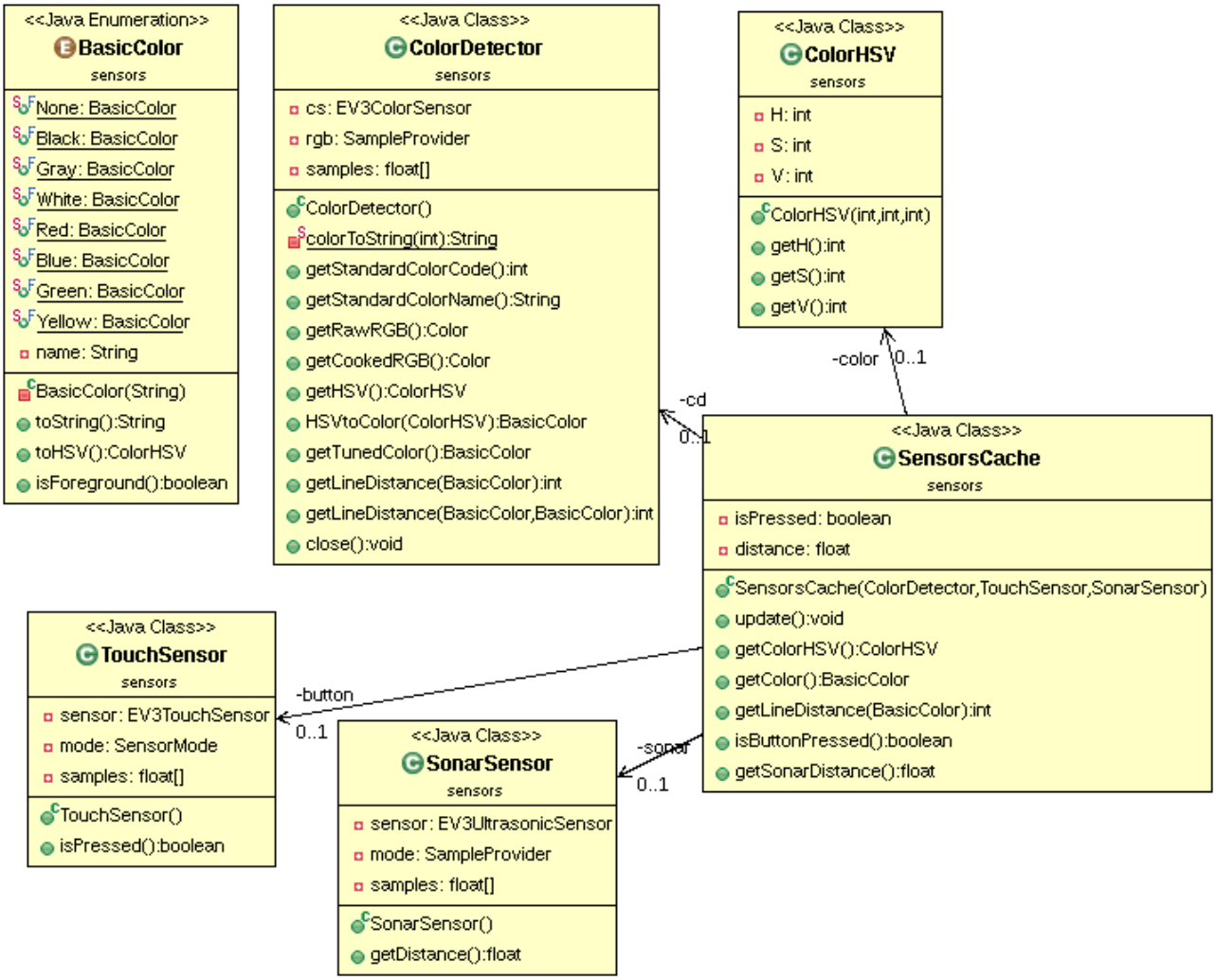
Package main:



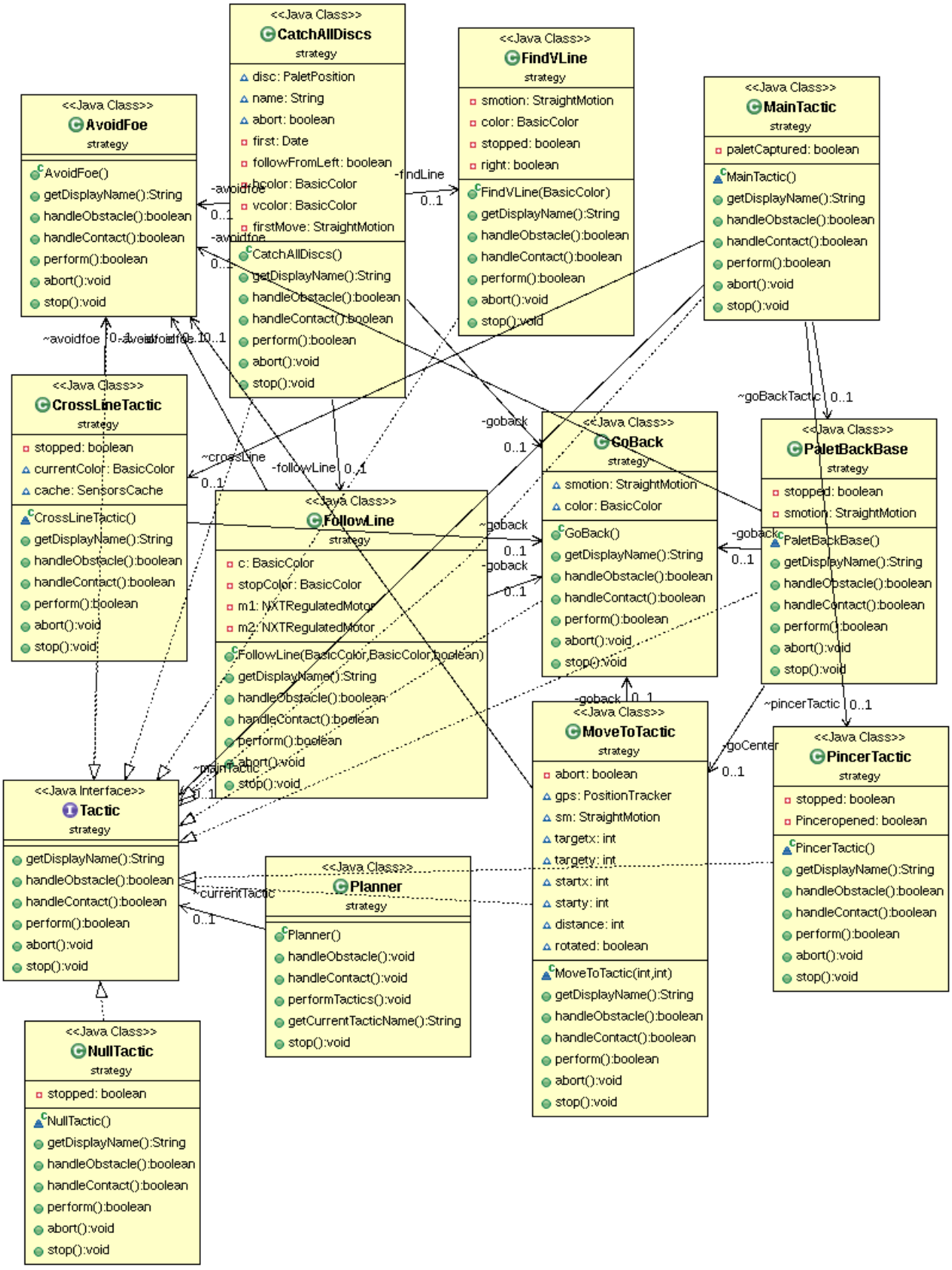
Package motor:



Package sensors:



Package strategy:



III. Plan de tests

1. Capteurs

a) Capteurs de couleurs : pour les couleurs, nous avons converti le capteur pour obtenir un système en HSV (Teinte Saturation Lumière). Nous avons calibré chaque couleur en fonction des valeurs qu'on obtenait de H, S et V.

b) Capteurs de palets : il s'agit seulement du bouton pression situé sous la tête du robot.

c) Sonar : les tests ont été effectués avec nos mains placés devant les yeux du robot. On a fait en sorte d'indiquer la distance captée.

2. Stratégies de base

- Suiveur de ligne : pour le suiveur de ligne, nous avons testé notre robot en lui indiquant la couleur de ligne à suivre ainsi que la couleur à laquelle il devait s'arrêter. Nous avons modifié les paramètres de vitesse des moteurs de chaque roue afin que Yeti aille le plus droit possible. Nous devions choisir une vitesse suffisamment acceptable pour aller vite mais pas trop pour pouvoir quand même s'arrêter.
  Dans notre cas, nous devions paramétrer deux vitesses : une vitesse de base permettant de faire avancer Yeti, et une vitesse qui permet de redresser Yeti sur la ligne lorsqu'il s'écarte trop de la ligne (on le calcule grâce à un pourcentage de couleur grise et de la couleur actuelle). Tout d'abord, les paramètres 280 pour la vitesse de base et 120 pour la vitesse de redressement paraissaient trop lentes, alors nous avons essayé à 560 et 240. Ces vitesses étaient trop fortes mais elles fonctionnaient.

- Eviter l'adversaire/l'obstacle : nous avons testé tout d'abord la précision du sonar qui envoie les ondes et si elles reviennent. Nous avons remarqué que si celui-ci indiquait infini s'il n'y avait rien devant lui mais aussi s'il y avait quelque chose juste devant lui. Avec ces données nous avons implémentés les deux stratégies afin qu'il évite un obstacle et qu'il ne reste pas sans rien faire au cas où. Pour éviter un adversaire, nous avons testé en mettant un obstacle avec la stratégie du suiveur de ligne. Il l'a bien évité et dépassé. Pour faire demi-tour si on s'est pris un obstacle ou un adversaire, nous avons essayé de la même manière.
  Cependant, en mettant les deux stratégies en combinaison avec les autres aucunes ne s'effectuait correctement, on les a donc enlevés lors de la démonstration.

- Rentrer dans la base : nous avons, premièrement, testé si le robot détecte un palet, c'est-à-dire fermer les pinces une fois un palet touche le capteur, une fois c'était bon nous avons ensuite pensé à une stratégie de rentrer dans la base, vu que le premier qui dépose un palet dans l'autre camp durant la compétition aura des points de bonus, nous avons développé cela en allant en diagonale une fois on récupère le premier palet. Dans notre cas, nous avons eu un problème car il fallait paramétrer deux cas :
  1) quand on commencer du coté bleu du terrain : ou il fallait utiliser des cordonnées positives.

2) où on commence de l'autre côté et il fallait utiliser des cordonnées négatives. Sinon, le robot se perd et tourne en rond.

Après cela, on a testé si le robot rentre à la base de l'adversaire pour tous les palets qu'il récupère et pas à notre base en lui indiquant qu'il fallait faire un demi-tour à chaque fois qu'il récupère un palet.

3. Stratégies globales

- Attraper tous les disques : Dans cette classe, nous avons rassemblé toutes nos stratégies de base afin de monter une stratégie qui récupère le plus de palets possibles. Cette classe a beaucoup évolué. Tout d'abord, il a fallu répertorier les positions précises de chaque croisement de ligne, i.e. l'emplacement de chaque palet dans un monde idéal où ils ne changeraient pas de place. Pour cela, nous avons utilisé le tachymètre du robot. Nous n'avions au départ calculer qu'une seule largeur et longueur d'un rectangle. Mais suite à des imprécisions remarquables, nous avons décidé de prendre mesure de chaque largeur et chaque longueur de ligne grâce au tachymètre. (Cf. Journal de bord dans l'annexe)

  La première version de CatchAllDiscs consistait à utiliser la position des palets dans la classe PaletPosition et ordonner à l'ordinateur d'aller vers le palet le plus proche. Comme les problèmes d'imprécisions persistaient, nous avons décidé de changer de méthode et d'utiliser le suiveur de ligne. Il a fallu alors moins considérer les coordonnées des palets que les lignes verticales sur lesquels ils se trouvaient.

  Avec cette nouvelle méthode, nous avons réussi à capturer jusqu'à 4 palets lorsque Yeti était seul sur le terrain. Cependant, nous n'avons pas réussi à faire amener le robot sur la ligne noire pour capturer plus de palets. En effet, lorsque le robot devait chercher le palet le plus proche se trouvant sur la ligne du milieu, il était complètement perdu.

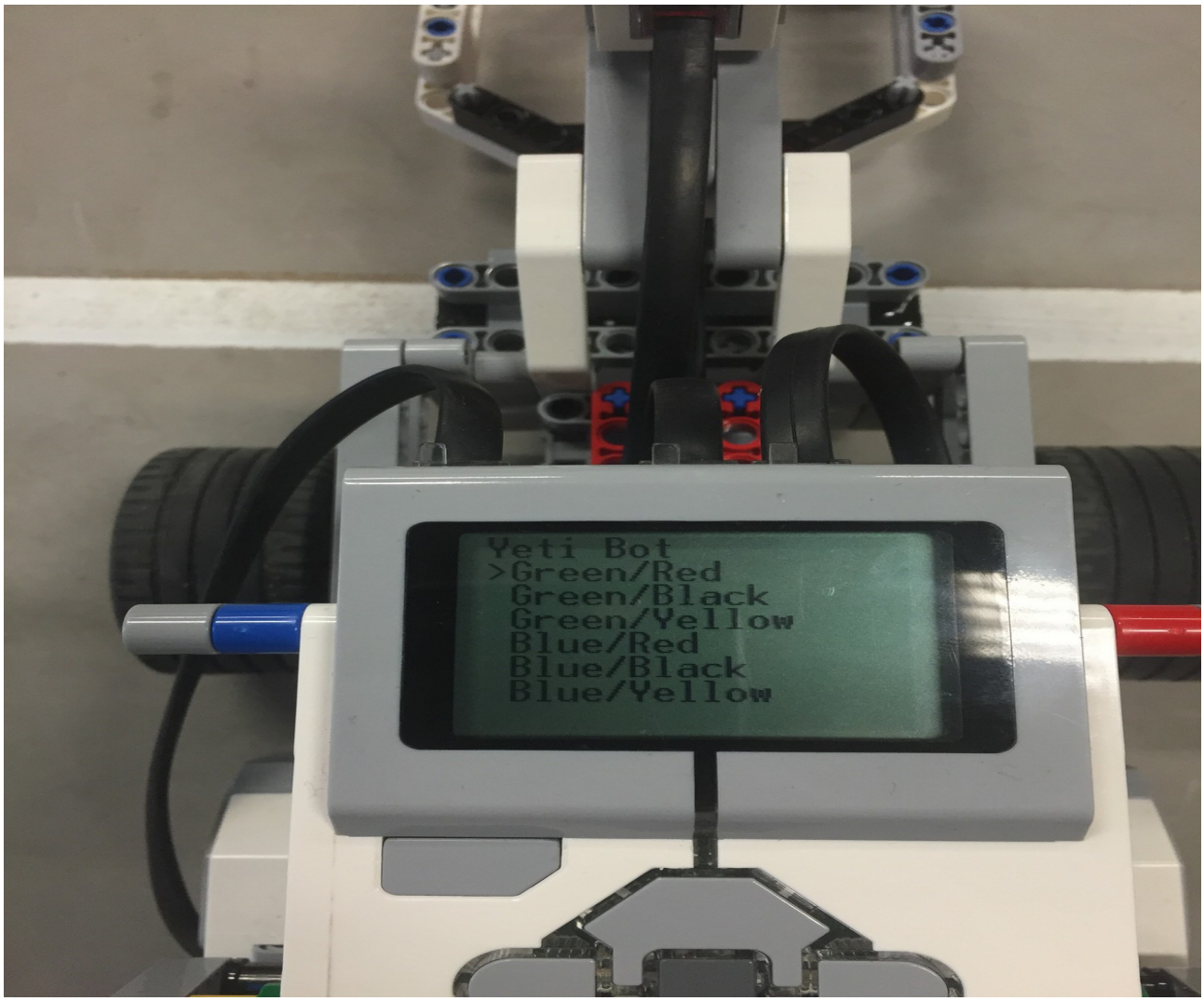
Conclusion

Ce projet nous a appris à avoir un début d'aperçu de ce que représente l'intelligence artificielle. Notamment, en ce qui concerne le cycle d'exécution du robot. Dans notre code, pour gérer les conflits entre les codes, il a fallu mettre des délais d'attente afin d'obtenir un comportement convenable du robot. Nous avons conscience de ne pas avoir fait d'intelligence artificielle à proprement parler mais en vue de la compétition, nous avons tenté d'aller au plus simple.

Pour la compétition nationale qui aura lieu à Grenoble, nous allons devoir améliorer les fonctionnalités suivantes :

- Récupérer les palets du milieu et ainsi en prendre le maximum possible,
- Eviter l'adversaire, le mur,
- Faire un lancement de programme beaucoup plus rapide, (nous avons déjà un affichage pour déterminer l'emplacement de départ du robot, mais à chaque fois que l'on reprend le robot pour le relancer, le programme met du temps à s'exécuter)
- Gérer la mise à jour de la position lorsqu'il croise la ligne noire car Yeti doit tenir compte de la verticalité ou non de la ligne noire qu'il rencontre.

Annexes :


Yeti Bot
>Green/Red
Green/Black
Green/Yellow
Blue/Red
Blue/Black
Blue/Yellow

Journal de bord

29/01/2016 : montage du robot en lego + installation de l'environnement leJOS

05/02/2016 : flash de la microcarte SD pour installer leJOS sur le robot nommé YETI

objectifs pour la prochaine séance :

- Analyser les capteurs
- Se documenter pour le moteur
- Réfléchir au cdcf

12/02/2016 : mise en place de la structure de base du programme
Établissement du CDCF
Répartition des tâches
Capteurs ok

19/02/2016 : mise en place d'une tactique qui permet d'ouvrir les pinces au début de la partie et les fermer lors de la détection d'un palet.

Tactique pinces ok
Tactique suiveur de ligne commencé

Objectifs : EE : suiveur de ligne : faire avancer le robot plus vite (mais pas trop pour qu'il puisse s'arrêter)
Ajouter un argument GAUCHE/DROITE

Younes : Tactique relâcher le palet

Timothée : commencer code GPS

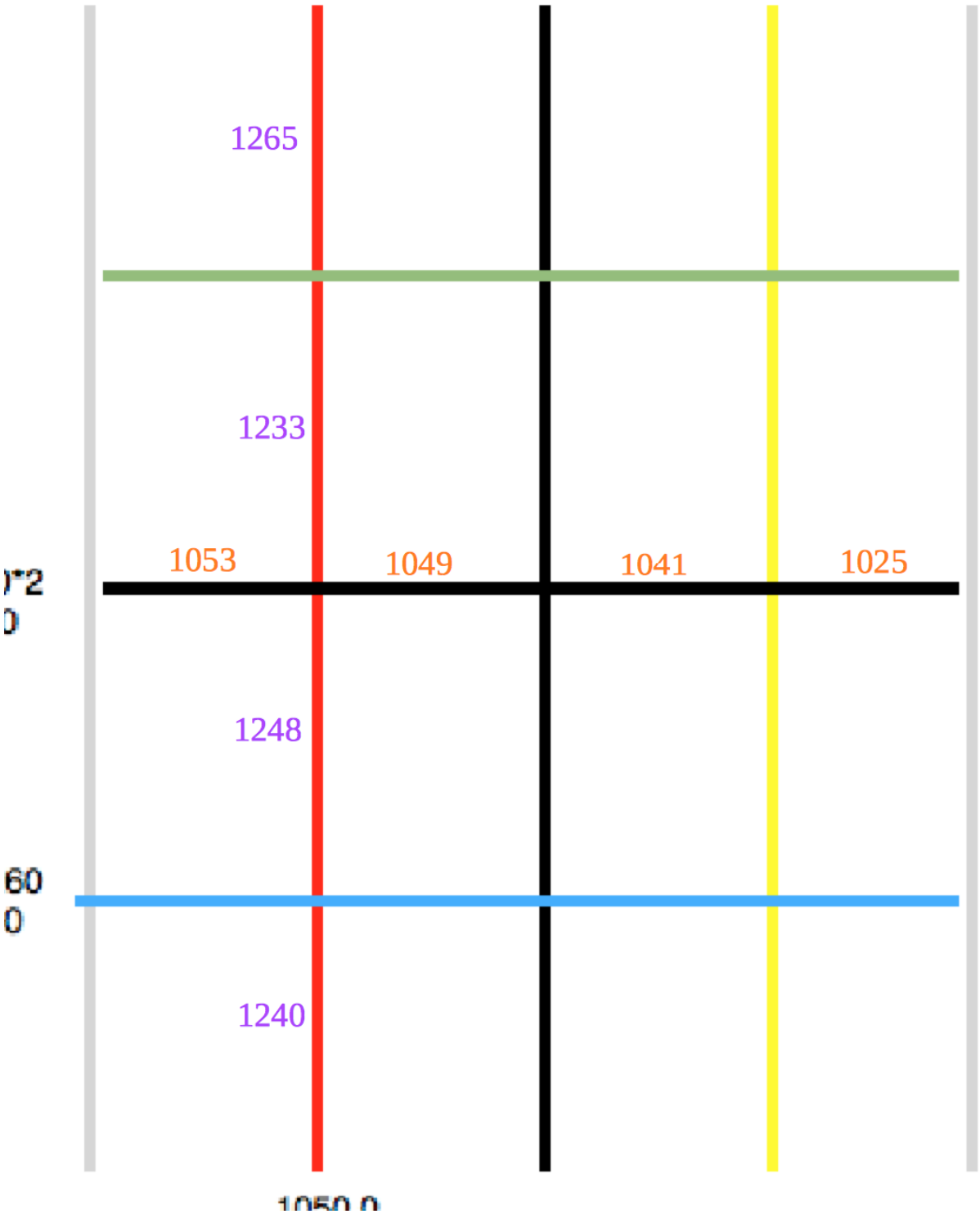
26/02/2016 : mise en place d'une tactique qui permet d'emmener les palets dans les camps adverse
: Amélioration followline pour aller plus droit, plus rapide

Objectif séance prochaine :
Elodie : navigation sur plateau
Emma : éviter l'ennemi et les obstacles
Timothée : gestion tactique
Younes : mouvement du robot vis-à-vis du palet

4/03/2016 : classe globale PaletPosition permettant de savoir quel est le palet libre le plus proche du robot
AvoidFoe fini
Améliorer goBack
Améliorer capteur de couleur
Travail sur GPS

11/03/2016 : Amélioration de PaletPosition
Amélioration du GPS
Implémentation de la tactique MoveTo

Système de coordonnées sur le terrain :
Longueur des cases en violet // Largeur des cases en orange



L'origine (0, 0) se trouve dans le coin de croisement des lignes BLEU/ROUGE

Idées pour plus tard : gérer les changements de positions des palets sur le plateau

Conversion unité Yéti / cm : 1cm ~ 21 unités

18/03/2016 :

Timothée & Younes : amélioration de PaletBackBase

Elodie : amélioration de PaletPosition et implémentation de CatchAllDiscs stratégie pour récupérer tous les disques sur le plateau

Emmanuelle : amélioration de GoBack

Objectif **25/03/2016 :** faire marcher CatchAllDiscs
Optimisation de CatchAllDiscs et PaletBackBase

1er avril : quand yeti arrive devant un palet, s'il ne le trouve pas il abandonne le palet et le considère comme pris
Modification de paletbackbase
Précision dans les roues
Modification de goback

Prochain RDV : lundi 4 matin ?
Objectif : mettre à jour les coordonnées en fonction des couleurs/ Utiliser les couleurs pour que Yeti avance droit
Aller jusqu'au bout de la stratégie CatchAllDiscs

8 avril 2016 : modification d'update
Créer une classe qui met à jour les coordonnées du robot quand il croise une ligne

Prochain rdv LUNDI 11 AVRIL 2016

Jeudi 14 avril :
Modification des stratégies CatchAllDiscs / PaletPositions
Désormais, le robot considère aussi les lignes verticales sur lesquelles se trouvent les palets pour avoir plus de précisions

Vidéo de tests du robot :
* Pour attraper le premier palet et le ramener à la base de l'en-but :
https://drive.google.com/file/d/0B8EiCLGlpGDed1Nyb196alFILTQ/view?usp=sharing

* Pour attraper 4 palets :
https://www.youtube.com/watch?v=_kx08sRN11s