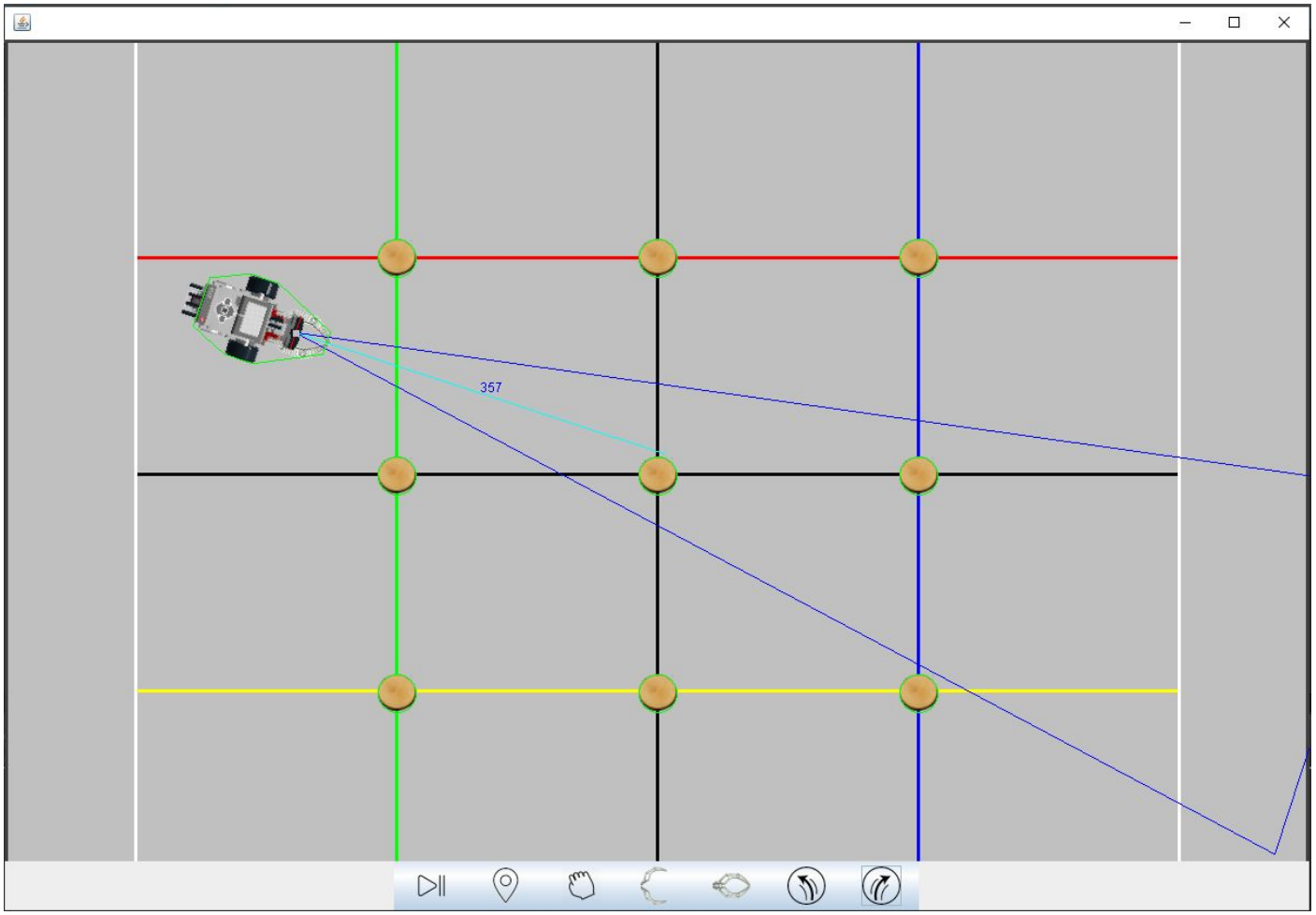
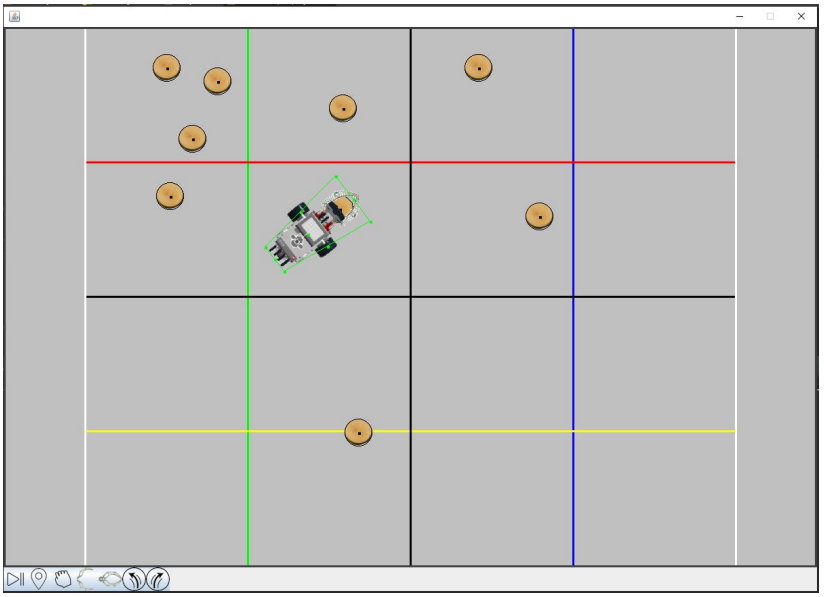
CAHIER DES CHARGES

Projet de modélisation graphique d'un robot LEGO ramasseur de palets



Évolution des bords du robot en fonction de l'ouverture des pinces, ci-contre tels qu'ils étaient représentés avant.



FLEURY Pierre **11716306**
GATTACIECCA Bastien **11808782**

L3 MIASHS
Version : 4.0
20/12/2020

Table des matières

Introduction

Ce projet se construit en parallèle du cours d'initiation à l'Intelligence Artificielle où l'objectif est de programmer un robot LEGO (physique) qui doit ramasser un maximum de palets pour le confronter à d'autres robots sur un plateau. Notre projet consiste à programmer une application graphique qui a pour objectif de modéliser au mieux des stratégies implémentées à un robot LEGO ramasseur de palet. Ainsi, elle doit permettre à son utilisateur de tester différentes stratégies sur un robot pour tester son efficacité. mais on peut également étendre l'utilité d'une telle application en implémentant par exemple la même stratégie sur les deux robots, ou autre. Ce document explicite les points importants à la réalisation d'un tel projet.

1. Contexte

Ce projet s'inscrit dans le cadre d'un cours durant le premier semestre de Licence 3 Mathématiques et Informatique appliqués aux Sciences Humaines et Sociales (MIASHS). Aucun sujet ne nous a été imposé. Nous avons décidé de nous inspirer d'un cours existant et des objectifs de ce dernier d'une part. D'autre part, nous choisissons d'utiliser nos compétences en programmation orientée objet pour nous permettre de développer de nouvelles compétences autour de ce sujet.

2. Présentation de l'équipe

Membres : FLEURY Pierre & GATTACIECCA Bastien
Tuteur : LEMAIRE Benoit

3. Description de la demande

Différents besoins ont été exprimés pour la réalisation du projet.
Il nous est demandé de fournir un effort pour l'organisation et la méthode vis-à-vis de la gestion de projet agile. Cependant, le résultat du travail rentre aussi dans les attentes auxquelles nous devons répondre.
Benoit Lemaire est le tuteur de notre projet et dans le cadre d'une gestion agile, il est nécessaire de privilégier nos interactions avec lui. Elles se traduisent par de courts rendez-vous mais réguliers (cf. phases itératives dans l'échéancier) pour des mises au point, des conseils ou des questions concernant la modélisation, etc. Nous mettrons en place un support pour montrer l'avancement du code à chaque mise à jour. Une mise à jour se décrit en terme de "nouvelle fonctionnalité" ajoutée. Nous lui transmettrons également un échéancier et un cahier des charges qui va être mis à jour au fil de l'avancement du projet.
Le projet doit faire l'objet d'un rapport qui sera présenté devant un jury.

4. Les objectifs

Le cours qui nous a inspiré repose sur du développement pour un système embarqué, il y a donc un objet physique (le robot) qui dépend de son environnement et des lois physiques qui l'entourent.

De notre côté, l'objectif est de reproduire au mieux ce robot, son environnement et ces lois de manière graphique afin de construire le meilleur modèle. Dans un premier temps, le but est de programmer les actions du robot graphique dans son environnement de manière théorique. Tout au long du projet, notre objectif sera d'implémenter à notre robot graphique (notre modèle, de manière plus générale) de nouvelles fonctionnalités pour que son comportement soit le plus proche de celui d'un robot physique. Le but étant de pouvoir simuler ces comportements lorsque les robots se confrontent lors de la compétition. Il est alors possible d'évaluer le modèle et donc de savoir quelle stratégie est la plus efficace.

L'objectif pédagogique ici est d'acquérir des compétences en développement graphique, de modélisation, mais aussi de gestion de projet informatique. Un dernier objectif est de reprendre les carnets de bord, bilans et auto-évaluations du projet de l'année dernière pour anticiper les imprévus traversés ou les erreurs commises.

Un objectif spécifique est d'empêcher l'utilisateur d'avoir accès (via la visibilité ou bien l'accessibilité de certains attributs, méthodes, ou classes) à certaines données utilisées pour coder les bases du programme auxquelles l'utilisateur n'aurait jamais eu accès dans la réalité. Par exemple, sa position (x,y) sur le plateau. Les seules données accessibles depuis l'environnement pour un robot sont celles perçues par l'ensemble de ses capteurs.

5. Produit du projet

- Un plan de développement pour définir les bases théoriques.
- Un diagramme UML, que nous mettrons à jour au fil de l'avancement.
- Le rendu sera une application en langage Java (très certainement un exécutable Jar avec un dossier pour les images utilisées).

6. Les fonctions du produit

- Le plan de développement doit contenir l'ensemble des signatures de méthodes (d'instance et de classe), des attributs de toutes les classes, avec leur documentation respective.
- Le diagramme doit présenter la généricité de notre code de base. On doit pouvoir facilement ajouter un robot, placer des palets dans l'environnement, instancier plusieurs environnements indépendants à la fois.
Ainsi, un tel code doit permettre d'éventuelles modifications si les règles du jeu étaient amenées à être modifiées. Comme modifier la taille du plateau ou encore changer les capteurs d'un robot.
- Coder les bases de l'application, comme :
    - Définir les images du plateau, du robot, des pinces, des palets ...
    - Ajouter à un robot ses capteurs et ses moteurs ;
    - Permettre aux capteurs de récupérer de l'information ;
    - Permettre aux moteurs de déplacer le robot ou les pinces ;
- Les fonctionnalités que va implémenter le programme :
    - Interaction physique entre les objets ;
    - Marge d'erreur des moteurs ;
    - Marge d'erreur des capteurs ;
    - Gérer les collisions ;
    - Gérer l'accessibilité et la visibilité du code ;
- Un rapport

A ce jour, voici les fonctionnalités que nous avons implémentées ; et celles que nous prévoyons dans la semaine qui suit.

Date	Résumé de l'itération, démonstration en vidéo	Annonce de la prochaine itération ; feedback de la précédente
22/10/2020 -> 05/11/2020	**Prise en compte de l'interaction physique entre tous les objets du plateau.** Le robot peut désormais entrer en contact avec les palets et les déplacer, les saisir avec les pinces. **Gestion des collisions** Deux palets ne se superposent pas	**Prendre en compte la marge d'erreur des trois moteurs, (vitesse, accélération)**

	(démonstration ici)	
05/11/2020 -> 12/11/2020	**Améliorations de la gestion des collisions** grâce à un changement d'algorithme et de la forme d'un robot **Calcul de la vitesse et de l'accélération** d'un robot physique en prenant en compte la variabilité et moteurs concernés (en exploitant cette vidéo que nous avons faite : ici)	**Moteur des pinces** : pas de prise en compte de variabilité (constante pour tous les robots de l'appli?) **La variabilité des trois capteurs + qualité des capteurs** (ajouter un balancier pour le capteur tactile, permettre au capteur à US de détecter des objets, marge d'erreur du capteur de couleur)
19/11/2020 -> 20/12/2020	**Améliorations des performances du code** grâce à l'implémentation d'un nouveau package (codé par nous-même) : utils ; qui permet d'utiliser des formes géométriques. **Améliorations de la gestion des collisions** qui prend maintenant en compte la position des deux objets mais aussi la vitesse à laquelle ils se percutent. **Le fonctionnement des trois capteurs** a été finalisé et ils fonctionnent comme prévu : - le capteur tactile renvoie true lorsqu'un objet appuie dessus - le capteur à ultrason regarde dans un cône de 20° et renvoie la distance de l'objet le plus proche - le capteur de couleur renvoie la couleur sous le "nez" du capteur **La variabilité des trois capteurs** a été implémentée grâce aux données récupérées depuis le robot physique. Le code RStudio pour l'étude stats a également été fini.	**Finaliser JavaDoc** **Visibilité des classes et des méthodes pour l'utilisateur final** **Optimisation du code** Pas de prochain rendez-vous, cette itération consisterait essentiellement

7. Contraintes

7.1 Contraintes de délais

Livrable	**Dernière mise à jour :**
Échéancier	05/10/2020
Cahier des charges	20/12/2020
Diagramme UML de classe	20/12/2020
Plan de développement	12/10/2020
Rendu du code	Début Janvier

7.2 Contraintes logicielles et matérielles

La difficulté principale repose sur la fiabilité de la reproduction d'un objet qui interagit dans un environnement complexe. En effet, il est nécessaire de produire des tests aussi bien logiciels que matériels pour calibrer le premier par rapport au deuxième. Par exemple, la variabilité des capteurs du robot physique doit aussi être prise en compte dans le programme.
L'application est dédiée, donc adaptée, aux règles et contraintes du projet d'IA. Ainsi, le règlement de la compétition écrite par D. PELLIER s'applique également à notre projet (*disponible ici* cf **10. Références**). A titre d'exemple, la taille du plateau, le traçage des lignes, la taille du robot, les capteurs du robot ... sont imposés.
Aucun cours en programmation concernant les méthodes graphiques ne nous a été donné. Le groupe n'a aucune connaissance dans ce domaine.
Une échelle de 1 cm pour 4 px sera prise. Ainsi la fenêtre de l'appli ne devra pas être redimensionnable puisqu'à priori elle n'affiche que le plateau. Cela donnerait une fenêtre de 1200x800 car le plateau fait 3m x 2m.

Il s'agit de la première gestion de projet en informatique, on va devoir assimiler de nouveaux outils pour avancer le projet. Par exemple GitHub pour partager le code ou pour créer un diagramme UML.
Vous trouverez ci-dessous le lien GitHub de ce projet. Il est possible via ce lien d'accéder à la dernière version du code de l'application graphique : https://github.com/fleuryP/ProjetJava.git

L'accès à du matériel adéquat est difficile en particulier à cause de la situation actuelle. Les mesures et tests sur le robot physique sont donc réalisés avec les moyens dont nous disposons.

7.3 Autres contraintes

Si la situation sanitaire venait à empirer, les cours en distanciel ne favoriseraient pas le bon déroulement du projet.
La situation sanitaire ayant empiré fortement à la date du 29/10/2020, nos cours se déroulent à présent en distanciels et il n'est actuellement pas possible pour le groupe de se réunir. Néanmoins, le fait que nous soyons deux dans ce groupe facilite la communication et le projet reste d'actualité. Les réunions avec le tuteur seront organisées via des plateformes collaboratives telles que Zoom ou Discord probablement. L'accès au robot reste inchangé étant donné qu'on a pu le récupérer.

8. Déroulement du projet

8.1 Planification

L'échéancier explique la phase de planification.
Chaque document à fournir illustre la fin d'une grande phase de ce projet. Voici comment nous prévoyons de planifier notre projet, en distinguant :
- la création d'un échéancier ;
- la définition des objectifs et des besoins dans le cahier des charges (mis à jour) ;
- la conception d'un diagramme de classe (mis à jour) et d'un plan de développement ;
- des phases itératives ; nous intégrons dans chacune d'elles :
    - la recherche d'une nouvelle fonctionnalité ;
    - un rendez-vous avec le tuteur pour, d'une part mettre à jour notre avancement sur la/les précédente(s) fonctionnalité(s), et d'autre part pour discuter de celle à laquelle nous avons réfléchis ;
    - faire une série de tests et/ou de mesures selon la nature de la nouvelle fonctionnalité ;
    - implémenter cette nouvelle fonctionnalité dans le programme ;
    - mettre à jour en conséquence le cahier des charges et le renvoyer, pour préparer le prochain rendez-vous ;

- la rédaction du rapport ;
- l'oral de la soutenance.

8.2 Ressources

Les deux membres qui composent l'équipe sont les seuls à contribuer au développement du projet. Il nous est possible de contacter plusieurs enseignants-chercheurs travaillant dans le domaine de la modélisation computationnelle.
Par ailleurs, l'accès à un robot Lego est facilité par le cours d'IA.
Des outils en libre accès qu'on a appris à utiliser ou que l'on va devoir apprendre à utiliser :

- Eclipse IDE pour la programmation Java ;
- Discord pour la com et le partage de fichiers ;
- Google Drive pour le partage de fichier ;
- RStudio ou Excel pour traiter les données statistiques ;
- GitHub pour le partage de code ;
- Diagrams.net pour faire le diagramme UML ;

9. Limites

Certaines actions sont difficiles à modéliser ; si par exemple deux robots se rentrent l'un dans l'autre, cela exige de prendre en compte de nombreux paramètres. On admettra nécessairement une marge d'erreur pour chaque itération que l'on justifiera durant la soutenance.
L'application n'a pas d'interface utilisateur. L'utilisateur doit avoir des compétences de base en Java pour pouvoir récupérer (en instanciant ou héritant) les méthodes auxquelles il a accès (comme déplacer, tourner le robot, récupérer la distance du capteur à ultrasons, etc).

10. Références

- règlement : https://lig-membres.imag.fr/PPerso/membres/pellier/doku.php?id=teaching:ia:project_lego