# I – Bilan

Afin de respecter le cahier des charges, nous avons modélisé notre programme selon l’UML fourni en annexe.

## Environnement

L’environnement comprend une arène qui peut être carrée ou circulaire. Le paramétrage est fait dans l’interface graphique, mais pour qu’il soit effectif en console, il doit être modifié dans le code.

## Obstacles et taches

Les obstacles et les taches peuvent être circulaires, carrés ou rectangulaires. Ces paramètres sont modifiables par l’interface graphique ou console.

## Robot

Le robot peut être carré, circulaire ou rectangulaire et contenir des capteurs à souhait. Pour simplifier l’exercice, nous avons créé une classe Roomba qui crée un robot aux caractéristiques du vrai roomba. Cependant il est possible de simuler un autre robot en créant une nouvelle classe.

## Capteurs

Le robot contient des capteurs qui sont de trois types : capteur de contact type poussoir, capteur de tache et capteur de distance laser. Ils peuvent être ajoutés au robot en modifiant le code, et peuvent servir dans la modélisation de comportement. Il est aussi facile d’ajouter un nouveau type de capteur à partir de la classe abstraite Capteur.

## Comportements

Nous avons implémenté 3 comportements différents pour le robot. Le premier dit IdiotVillage qui fait tourner le robot en rond indéfiniment. Le second est EvitObstacles. Celui-ci utilise les capteurs de contact et le capteur de tache pour suivre les murs et nettoyer les taches qu’il croise. Le dernier comportement est appelé Hazard(). Celui-ci utilise en plus des autres capteurs des télémètres lasers qui lui permettent de choisir une trajectoire plus efficace en évitant les obstacles et accéder à des zones plus difficiles d’accès.

L’intérêt est aussi de montrer que des comportements différents peuvent être implémentés et tirer parti des différents capteurs.

## Interfaces

2 interfaces ont été implémentés dans le programme ; l’une graphique et l’autre en console. L’interface console permet à l’utilisateur d’ajouter des taches et des obstacles dans une arène carrée de 4m de côté. Le lancement de la simulation affiche les positions du robot et les taches nettoyées.  
L’interface graphique permet une plus grande liberté de paramétrage de la simulation.

Tout d’abord, à l’ouverture de la fenêtre, on peut rajouter des obstacles ou des taches de forme variées en sélectionnant le composant correspondant dans le menu de gauche puis en dessinant dans l’arène. Une fois ces éléments déposés, on peut les supprimer en les déplaçant hors de l’arène ou sur le robot. Ces opérations se font par click&drag. L’interface permet aussi de choisir une pièce carré ou circulaire. Les configurations de pièces peuvent être sauvegardées en cliquant sur « Save », et chargées à partir de « Load ». Le bouton « Init » permet de réinitialiser la simulation. Le robot s’anime lors du clic sur Start et est stoppé grâce au bouton stop. A tout moment la pièce peut être reconfigurée avec l’ajout d’obstacles ou de taches.

L’interface permet aussi d’avoir des retours du robot, puisque l’on peut voir sur la partie gauche l’état des capteurs de contact situés à l’avant, et les distances calculées par les télémètres lasers.   
Le bouton Trace permet de tracer la trajectoire du robot afin de voir les zones parcourues.

## Fonctionnement de l’interface

Pour être facilement extensible et modifiable, l’interface doit être indépendante de la simulation en elle-même. Elle est donc dépendante de la simulation qui est lancée en thread pour tourner parallèlement à l’interface.

La modélisation orientée objet a été pensée pour pouvoir étendre les caractéristiques de la simulation assez facilement.