TME SOLO - 1h45

Dans ce TME SOLO, vous pouvez (et devez) réutiliser tout ce que vous avez fait durant le projet. Lorsque l'on parle de centimètre dans l'énoncé, il s'agit d'une unité de mesure constante quelconque dans votre arène simulée.

Avant de commencer le TME SOLO, créez une nouvelle branche dans votre dépôt `github` qui porte votre nom. Vous travaillerez uniquement dans cette branche. A la fin du TME, vous enverrez un email à `[nicolas.baskiotis|olivier.sigaud]@sorbonne-universite.fr` qui contiendra l'adresse du `github`, le nom de la branche et la liste des questions traitées **en indiquant obligatoirement les questions auxquelles vous avez répondu et quels fichiers ont été modifiés pour ces questions**.

En dernier recours, si vous n'arrivez pas à sauver votre travail dans la branche, vous enverrez par email votre code.

Dans un fichier `tmesolo.py` à la racine de votre dépôt, vous regrouperez les fonctions `qX.Y()` qui permettent d'exécuter le code correspondant aux questions `X.Y` ainsi que les affichages graphiques si vous en avez.

Les parties sont indépendantes, vous pouvez les traiter dans l'ordre que vous voulez. Si votre plateforme ne dispose pas de certaines fonctionnalités, vous pouvez les supposer implémentées.

Exercice 1 – Manipulations simples

Q 1.1 Fabriquez une simulation avec 3 obstacles alignés (de forme quelconque, carré, rectangle ou rond), un au centre, un en haut au milieu, et un en bas au milieu. Placez votre robot dans le coin inférieur gauche.

Q 1.2 Faites une stratégie qui bouge le robot horizontalement vers la droite (à partir du coin inférieur gauche) et qui quand il s'approche trop près d'un mur ou d'un obstacle, fait faire un demi-tour au robot. La stratégie boucle 10 fois (elle fait 10 demi-tour et s'arrête).

Q 1.3 On veut équiper le robot d'un crayon qui peut dessiner sa trace quand il est abaissé (et ne dessine rien quand il est levé) d'une couleur bleue. Ajoutez une commande `dessine(b)` avec b booléen au robot et modifiez le nécessaire dans votre affichage de telle façon qu'à partir du moment où `dessine(True)` est appelée, la trace de votre robot est affichée dans la simulation. Un appel à `dessine(False)` permet de désactiver la trace dans l'affichage graphique.

Q 1.4 Le robot a en fait la possibilité de tracer selon deux couleurs, bleu ou rouge. Ajoutez une commande `rouge()` et une commande `bleu()` au robot. Lorsque la commande `rouge()` est appelée, la couleur de la trace passe au rouge ; si c'est `bleu()`, alors la couleur passe au bleu. Cela ne déclenche pas forcément le tracé ! Si on exécute `dessine(False)` puis `bleu()`, alors il n'y a pas de trace dessiné, mais la prochaine fois qu'on appelera `dessine(True)`, la couleur sera bleue.

Q 1.5 Donnez les trois stratégies : `StrategieBleu`, `StrategieRouge`, `StrategieInvisible` qui permettent de déclencher la trace bleue, la trace rouge, et la trace non visible (ces stratégies ne font rien d'autres que de changer la couleur de la trace et de déclencher la trace). Combinez-les avec la stratégie de la 2ème question : quand le robot bouge vers la droite, il laisse une trace rouge, sinon une trace bleue. *Attention : le changement de couleur ne doit pas intervenir dans les stratégies qui font avancer ou tourner le robot, il ne doit intervenir que par l'intermédiaire des trois stratégies précédentes.*

Exercice 2 – Modification de l'environnement

On appellera votre premier robot la *souris*. Il aura toujours une trace bleue. On ajoute un deuxième robot dans votre simulation qui jouera le rôle de chat, qui aura une trace rouge et qui est de la même forme et même taille.

Q 2.1 Ajoutez un deuxième robot à votre simulation. Il aura un contrôleur indépendant du premier.

Q 2.2 Testez en faisant dessiner à la souris un carré du côté gauche de la simulation, et au chat un aller/retour haut bas.

Q 2.3 On suppose que le chat attrape la souris si le chat se trouve à une distance de moins de deux fois le diamètre de la souris (la distance entre les deux roues). La simulation doit s'arrêter lorsque le chat attrape la souris. Implémentez cette condition et testez-la (par exemple en alignant au départ le chat et la souris, en laissant immobile la souris et en faisant avancer le chat vers la souris).

Q 2.4 Le chat voit la souris si la souris est dans un angle compris entre -10° et 10° par rapport à son orientation. On suppose pour cela qu'il est muni d'un capteur `get_souris()` qui renvoie la distance et l'angle sous la forme d'un angle quand la condition est remplie, `None` sinon. Implémentez ce capteur.

Q 2.5 Implémentez une stratégie pour le chat qui lui permet d'attraper la souris le plus vite possible.

Q 2.6 Modifiez `get_souris()` afin de prendre en compte les obstacles (le chat ne voit pas à travers les obstacles).

Q 2.7 (Question ouverte) On suppose que la souris connaît à chaque instant la position du chat. Proposez une stratégie pour que la souris survive le plus longtemps possible.