# Modélisation et robotique

Exercices 11

## 1 Simuler Thymio

Le repère du simulateur est quelque peu étrange, mais c'est ainsi que cela fonctionne souvent à l'écran : l'axe des y est vertical mais croît vers le bas tandis que l'axe des x est inchangé. Par conséquent, le sens direct correspond à des rotations vers la droite.

**Question A. Simulation du capteur de proximité.** Le but de cette question est d'écrire une fonction du simulateur `get_prox()` qui retourne une liste de cinq valeurs simulant les capteurs de proximité du Thymio.

Les cinq capteurs de proximité avant du Thymio détectent les obstacle selon un faisceau serré assimilable à une télémétrie dans le simulateur robosim. La seule différence notable avec une télémétrie laser est que la portée de ces capteurs de proximité est beaucoup plus réduite. Ces capteurs sont disposés à 8 cm du centre de rotation du robot (le trou central entre le roues), et orientés selon des axes dont les écarts consécutifs sont de 20 degrés. La portée des capteurs est à peu près de 8 cm, c'est à dire 16 cm en partant de l'axe central du Thymio. Pour simuler une mesure prise par les capteurs de proximité, vous pourrez utiliser la fonction `telemetre_coords_list` de robosim qui sera plus rapide que de prendre 5 mesures de télémetrie (attention cette fonction prend un angle relatif au repère non au robot, il vous faudra donc également utiliser `orientation()`.

Vous utiliserez une variable globale `horizon` de valeur 50 pixels pour pour représenter la distance en dessous de laquelle les capteurs simulés seront capables de faire une mesure. Les mesures de télémétrie supérieures à la valeur de `horizon` seront remplacées par la valeur `horizon`.

Écrivez et testez la fonction `get_prox()`.

**Question B. Déterminer un cap.** Écrire une fonction `cap(x, y)` qui prend en entrée les coordonnées d'un point dans la simulation et retourne l'angle qu'il faut prendre (relativement au repère) pour orienter le robot vers ce point. Vous aurez besoin des coordonnées actuelles du robot et donc de la fonction `position()`, de la fonction `math.atan2(y, x)` (très connue en informatique) et d'une conversion des radians vers les degrés pour laquelle nous vous recommandons de créer une fonction (ainsi que la fonction réciproque qui convertit des degrés en radians).

**Question C. Tester le cap.** Pour tester votre fonction qui fixe un cap, orienter le robot vers la droite avec `set_heading(0)` puis demandez lui de prendre le cap du point (460, 460) et d'avancer de 5 pixels.

**Question D. Adoucir un changement de cap.** Les données sont comme à la question précédente : l'orientation initiale est l'angle 0 et on veut aller au point (460, 460). Mais on ne modifie le cap que lentement en prenant à chaque tour un peu plus la direction du cap fixé par le but de telle sorte que le nouvel angle soit égal à $\alpha$ fois le cap à prendre et $(1-\alpha)$ fois l'ancien cap, comme dans le programme suivant.

```
angle = 0; x, y = 460, 460; alpha = 0.1
for i in range(100):
    angle = alpha * cap(x, y) + (1 - alpha) * angle
    setheading(angle)
    av(5)
```

Testez simplement ce code. Vous pouvez essayer d'autres valeurs de $\alpha$.

**Question E. Éviter les obstacles.** Nous allons utiliser le capteur de proximité pour éviter les obstacles. Pour cela, créer une fonction `cap_evitement()` qui fait appel à `get_prox()` et combine les angles des capteurs selon les cinq valeurs obtenues pour calculer un angle d'échappement $\theta$, comme ceci :

$$\theta_{\text{échappement}} = \beta \times \sum_{i=0}^{i=4} \texttt{prox}[i] \times (40 - 20 \times i)$$

Cette formule donne un angle relatif au robot et elle aura tendance à l'orienter vers les directions dans lesquelles les capteurs ne détectent pas d'obstacle. Le facteur $\beta$ servira à normaliser la valeur de l'angle (on pourrait le prendre égal à l'inverse de la somme des valeurs des capteurs, mais il est plus intéressant ici de s'en servir pour augmenter l'angle d'échappement lorsque certaines valeurs des capteurs sont faibles). La fonction `cap_evitement()` retournera l'angle d'évitement.

**Question F. Combiner cap et évitemment.** Combiner cap et évitement en utilisant $\alpha$ fois le cap relatif et $(1 - \alpha)$ fois l'angle d'évitement dans un déplacement par pas de 5 en direction du point $(460, 460)$. Vous devriez obtenir de bons résultats avec $\alpha = 0.1$ et $\beta = 0.01$. Attention pour que cette combinaison ait un sens il faut que les angles soient exprimés dans le même repère (celui du robot ou le repère de l'écran). Vous pouvez rendre le cap relatif au robot lui soustrayant la valeur de `orientation()`.

**Question G. Optimisation.** Vous aller maintenant optimiser les valeurs de $\alpha$ et $\beta$ à l'aide de la méthode $1 + 1ES$ utilisée la semaine dernière. Pour cela créer une fonction prenant un paramètre une liste de deux valeurs entre 0 et 1 et servant à fixer la valeur de $\alpha$ et $\beta$. Cette fonction retournera une valeur qui sera d'autant plus faible que le déplacement jusqu'au point $(460, 460)$ selon l'algorithme précédent sera un succès. Vous pouvez par exemple faire le déplacement en un maximum de 200 itérations et interrompre la boucle dès que le robot est à une distance inférieure à 10 pixels du but puis retourner le nombre d'itérations effectuées plus la distance à laquelle le robot se trouve du but.