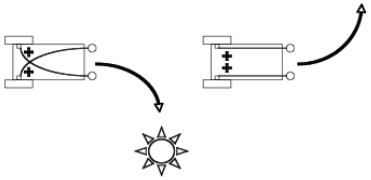
Robotique et comportements réactifs

Véhicule de Braitenberg, Architecture de subsomption, Boïds

Mise à jour : Mars 2017

Rendez-vous sur la page http://pages.isir.upmc.fr/~bredeche/ - onglet 3i025 pour télécharger le code source Python nécessaire pour faire les exercices ci-dessous (dépendances: Pygame, Matplotlib). Dans l'archive que vous récupèrerez, seul le fichier *multirobots.py* vous intéresse. Pour toutes les questions qui suivent, vous devrez faire une copie de ce fichier avant de commencer. Il existe une aide sommaire dans les commentaires au début du fichier, et le fichier contient tout ce qu'il faut pour vous aider à utiliser les fonctions utiles à la réalisation des questions ci-dessous. Passez un peu de temps pour vous familiariser avec le fonction **step(.)**.

Véhicules de Braitenberg [Braitenberg, 1984]

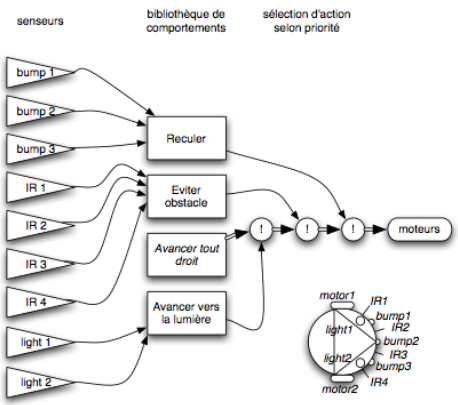


L'image ci-contre montre deux exemples de véhicules de Braitenberg. Dans ces exemples, les connexions reliant les entrées sensorielles (ici: deux photorécepteurs) et les sorties motrices (ici: les deux moteurs) sont soit excitatrices, soit inhibitrices.

En s'inspirant de ces deux exemples, on vous demande de programmer trois comportements: (1) comportement "aller vers les obstacles" (fichier *braitenberg_love.py*, à créer), (2) comportement "fuir les obstacles" (fichier *braitenberg_hate.py*, à créer), (3) aller vers les autres robot et (4) fuir les autres robots. Par défaut (i.e. en l'absence d'obstacles), les comportements commandent un déplacement en ligne droite.

Testez avec 1 seul agent, puis avec 10 agents (variable *nbAgents)*. Attention: les comportements (1) et (2) sont censés *ignorer* les autres robots.

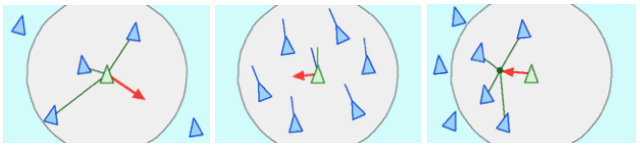
Architecture de subsomption [Brooks, 1986]



La figure ci-contre donne un exemple d'architecture de subsomption, composée de comportements et d'un processus de sélection d'action qui précise l'ordre de priorité des comportements en fonction de leur activation possible. Vous pouvez vous en inspirer pour répondre à la question suivante.

Dans l'exercice précédent, les comportements (3) et (4) ne permettent pas de gérer l'évitement d'obstacle. Pour remédier à cela, créez un fichier *subsomption.py* afin d'implémenter pour chaque agent une architecture de Subsomption disposant de trois blocs de comportement:
(a) Aller tout droit
(b) Evitement d'obstacle
(c)comportement d'évitement des robots
A vous de fixer les priorités entre comportements.

Comportements collectifs: les Boïds [Reynolds, 1984]



Le comportement des "Boïds" est controlé par trois règles: attraction, répulsion et orientation. Chaque règle s'applique en fonction de la distance aux voisins (distance au barycentre ou au plus proche -- au choix). Pour implémenter ces comportements, chaque agent doit pouvoir estimer la distance à ses voisins, ainsi que sa différence d'orientation.

Créez un fichier *boids.py*, et implementez le comportement des boïds pour 40 agents. Vous pouvez supprimer les murs. A vous de régler le seuil d'application de chaque règle.

Afin de gérer le cas particulier des murs, il est conseillé d'ajouter les comportements boïds dans une architecture de subsomption dont le comportement de plus haute priorité est l'évitement de murs.