



alphaai

L'EDUCATION À L'INTELLIGENCE ARTIFICIELLE

PROGRAMMATION CAMERA

INTRODUCTION

Le but de ce TP est de programmer AlphaI pour qu'il se déplace de manière autonome et non plus en mode « télécommandé » par l'élève comme dans le TP « AlphaI télécommandé ». Pour cela il faudra utiliser ses capteurs pour qu'ils puissent réagir à son environnement, notamment détecter les obstacles. Nous allons voir un exemple rapide avec l'ultra-son, puis passerons à la caméra pour organiser des courses de robots autonomes dans un circuit.

CONNAISSANCES REQUISES

- Programmation Python
 - Boucles *while* et conditions *if/elif/else*
 - Créer et appeler une fonction
- Ce TP est la suite logique du TP « AlphaI télécommandé » mais il n'est pas obligatoire de l'avoir effectué.

MATÉRIEL

- Un robot AlphaI par participant ou par équipe participante
- Un ordinateur par robot avec le logiciel AlphaI installé. Les ordinateurs doivent posséder une carte Wi-Fi ou Bluetooth pour communiquer avec le robot.
- Arène(s) où faire circuler les robots, si possible se présentant sous forme de circuit(s) pour l'organisation d'une course en fin de séance

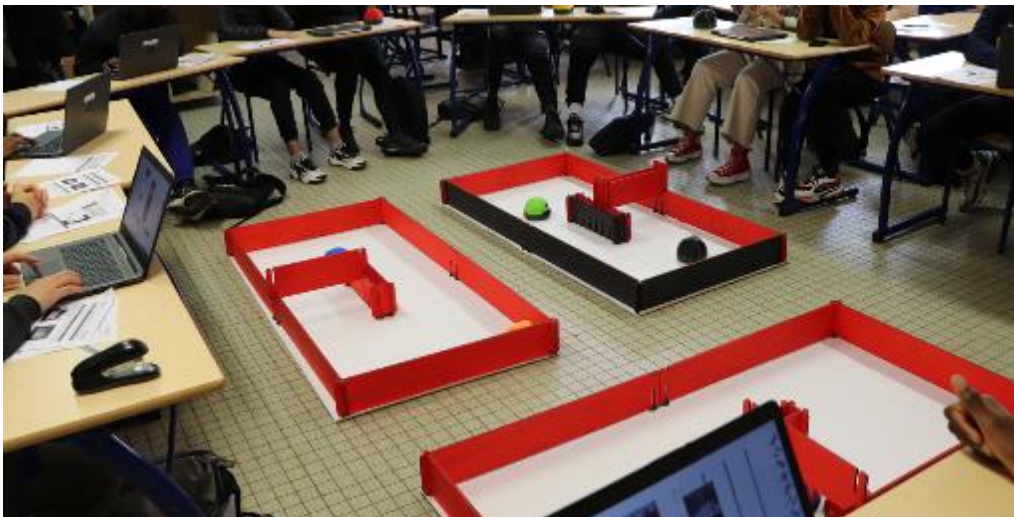


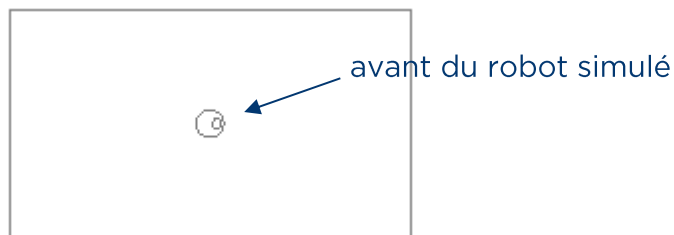
Figure 1 : multiples arènes « circuit de course »

PRISE EN MAIN DU ROBOT

Test de la librairie alphai pour contrôler le robot simulé

La librairie `alphai.py` permet de communiquer avec le robot en passant par le programme principal.

Lancez le logiciel en cliquant sur l'icône « **AlphaI** » sur le Bureau ou depuis le menu démarrer. En l'absence d'une connexion avec le vrai robot, un petit robot simulé apparaît en bas à droite.



Dans votre éditeur Python favori, créez un nouveau projet à l'emplacement de votre choix et configurez l'environnement Python pour qu'il contienne le module `alphai` en tapant dans la console IPython ou un terminal :

```
pip install alphai-api
```

Cette commande va télécharger et installer le module et ses dépendances.

Ouvrez un nouveau document python ou redémarrez la console, recopiez-y puis exécutez le code suivant :

```
from alphai import *  
  
motor(30, -30, 5)
```

Import de toutes les fonctions
de la librairie `alphai`

Si la première ligne du programme cause une erreur, c'est que la librairie `alphai` n'a pas été installée correctement : appelez le professeur pour demander de l'aide. Si la ligne `motor(30, -30, 5)` cause une erreur, c'est probablement que le logiciel `AlphaI` n'a pas été démarré.

S'il n'y a pas d'erreur, vous observez dans le logiciel le robot simulé tourner sur lui-même pendant 5 secondes.



La commande `motor(left, right, duration)` envoie les valeurs `left` et `right` aux moteurs gauche et droit du robot pendant la durée `duration`, après quoi il s'arrête. Les valeurs `left` et `right` doivent être comprises entre -50 et +50. Si la durée n'est pas précisée, alors le robot continue d'avancer jusqu'au prochain appel `motor(0, 0)`.

Vous trouverez à la fin de cette feuille de TP l'ensemble des fonctions incluses dans la librairie `alphai`. Consultez-les dès maintenant pour en prendre connaissance.

Connexion au robot AlphaI

- Allumez le robot (l'interrupteur se trouve en dessous). Il effectue un petit mouvement puis clignote en blanc une fois qu'il est prêt à recevoir une connexion.
- Notez le numéro du robot inscrit sur sa plaque du dessous.
- Choisissez si vous voulez vous connecter en Wi-Fi ou en Bluetooth (*nous recommandons la connexion **Bluetooth**, qui est préférable dès qu'il y a plus de 4 robots ou de nombreux réseaux Wi-Fi déjà présents dans la salle*).

Connexion Wi-Fi	Connexion Bluetooth
<ul style="list-style-type: none">- Connectez l'ordinateur au Wi-Fi du robot : cherchez le réseau Wi-Fi qui commence par ALPHAI et se termine par le numéro de votre robot : le mot de passe est identique au nom du wifi	<ul style="list-style-type: none">- Dans le logiciel, menu Connexions > Bluetooth, sélectionnez dans la liste le nom (numéro) de votre robot.- S'il n'est pas dans la liste, cliquez sur « Mon robot n'est pas dans la liste » et suivez les instructions. Votre robot est alors ajouté à la liste où vous pouvez le sélectionner.

- Dans le logiciel, cliquez sur le bouton « connexion »  pour vous connecter au robot.
Vous voyez apparaître son niveau de batterie  en bas à droite.

Exécutez à nouveau le même code que tout à l'heure : à présent, c'est le vrai robot qui tourne sur lui-même !

Il est possible et parfois utile d'effectuer la connexion dans votre script python plutôt qu'en cliquant « connexion » dans le logiciel, en utilisant les fonctions `connect_bluetooth(n)` (en remplaçant `n` par le numéro de votre robot) ou `connect_wifi()`, ainsi que `disconnect()` pour se déconnecter. Le code dans ce cas-là se présente ainsi :

```
from alphai import *  
  
connect_bluetooth(n)  
motor(30, -30, 5)  
disconnect()
```

Cette fois c'est le vrai robot qui tourne sur lui-même !

NAVIGATION AVEC L'ULTRA-SON

Découverte du capteur ultra-son

La fonction `get_distance()` (détails dans l'API en dernière page) renvoie la distance mesurée par l'ultra-son du robot (en centimètres). Attention à bien activer le capteur avec la fonction `set_distance(True)` ou elle ne renverra pas une distance mais la valeur `None`.

Testez cette fonction avec le vrai robot avec le code suivant :

```
set_distance(True) # active le capteur à ultra-sons
while True:
    d = get_distance() # récupère la valeur des ultra-sons du robot
    print("distance:", d)
```

En déplaçant le robot à la main, vérifiez que la distance mesurée diminue lorsque le robot est face à un obstacle. Vous pourrez aussi remarquer que malheureusement si le robot est devant un mur, non pas pile en face de lui, mais en diagonale, ce mur n'est pas détecté (la distance mesurée est beaucoup plus grande car les ultra-sons se reflètent sur le mur en diagonale sans revenir directement vers le robot).

Déplacement autonome avec l'ultra-son

A l'aide de la fonction `get_distance()`, définissez puis testez une fonction `ultrasound_race()` qui fait avancer le robot tant qu'il ne perçoit pas d'obstacle avec son ultra-son, et pivote à droite lorsqu'il détecte un obstacle.

Ne cherchez pas à avoir un comportement parfait (ce serait difficile), dès que vous avez un comportement où le robot sait tourner s'il est face à un mur et aller tout droit s'il n'y a pas de mur, passez à la suite.

NAVIGATION AVEC LA CAMERA

La caméra apporte une information beaucoup plus riche que le capteur de distance ultra-son, et permettra de prendre des décisions plus sophistiquées.

Découverte de l'image caméra

Commençons par afficher l'image de la caméra du robot dans le logiciel. Ouvrez l'onglet de paramètres « Capteurs » à gauche, et choisissez le mode de caméra « image 64x48 »

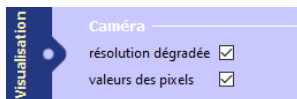


: l'image de ce que voit le robot apparaît.

Une image est une grille de pixels ; dans une image couleur comme c'est le cas ici, chaque pixel contient 3 valeurs : les intensités de rouge, vert et bleu dans ce pixel. Pour voir cela de manière plus précise, choisissez à présent le mode de caméra avec la résolution faible

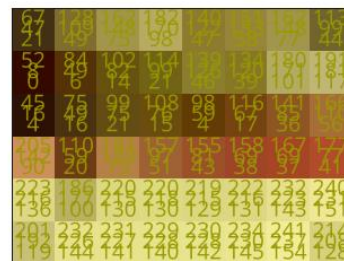
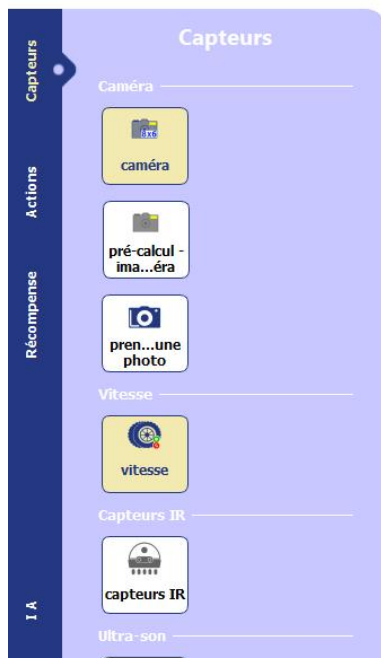


« image 8x6 » , et dans l'onglet « Visualisation » cochez « Caméra : résolution



dégradée » : une image très grossière apparaît, et les valeurs rouge/vert/bleu de chaque pixels sont affichées (valeurs entre 0 et 255).

AlphaJ, robot apprenant
Paramètres I.A. Outils Aide Développeur



Pour régler la résolution d'image dans le code Python plutôt que dans le logiciel, il faut commencer par initialiser la caméra, par exemple `set_camera("8x6")`. Ensuite on récupère l'image avec `get_camera()`. Cette fonction renvoie un tableau de dimension 3 sous la forme d'un objet « List ». La seule chose à savoir pour ce TP à propos de cet objet est comment

accéder aux valeurs des pixels : Si on appelle `image` cet objet renvoyé, on accède à la valeur du pixel de la ligne i , colonne j et dans le canal k ($k=0$ pour rouge, 1 pour vert, 2 pour bleu) avec `image[i][j][k]`.

Copiez et exécutez le code suivant, et vérifiez que les 3 valeurs affichées correspondent bien aux 3 valeurs rouge/vert/bleu dans le logiciel du pixel à la 5^{ème} ligne en partant du haut (indice 4 quand on commence à compter à zéro) et de la 4^{ème} colonne en partant de la gauche (indice 3).

```
from alphai import *

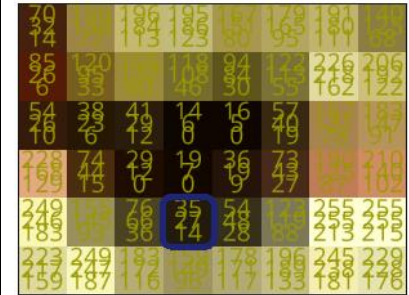
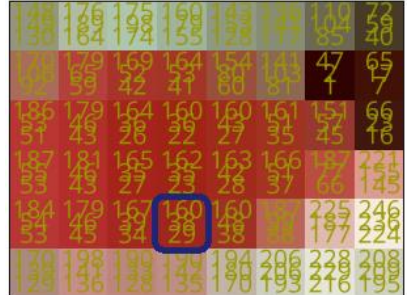
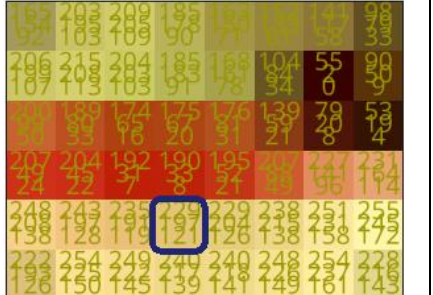
set_camera("8x6") # active la caméra à la bonne resolution

while True:
    image = get_camera() # récupère l'image de la caméra
    line = image[4]
    pixel = line[3]
    print('rouge', pixel[0], 'vert', pixel[1], 'bleu', pixel[2])
```

Premier déplacement autonome avec la caméra

Nous voulons que le robot effectue de manière autonome des tours de circuit. Si vous utilisez une arène fournie avec les robots AlphaI, le sol est blanc, l'enceinte extérieure est rouge, et l'îlot central est noir. Nous supposons que c'est le cas, et que vous voulez faire circuler le robot en tournant dans le sens des aiguilles d'une montre.

Commencez par écrire une fonction `camera_race()` pour prendre les décisions répétitivement, de la manière suivante (qui se base uniquement sur la couleur d'un seul pixel dans la partie basse de l'image) :

		
<p><u>Si</u> <code>image[4][3][0] < 100</code> (canal rouge sombre → le pixel est noir), pivoter à gauche à vitesse 25 pendant 0.5 seconde</p>	<p><u>Sinon</u>, si <code>image[4][3][1] < 100</code> (canal rouge lumineux et canal vert sombre → le pixel est rouge), pivoter à droite à vitesse 25 pendant 0.5 seconde</p>	<p><u>Sinon</u> (canaux rouge et vert lumineux → le pixel est blanc), aller tout droit à vitesse 40 pendant 0.5 seconde</p>

Testez ce programme sur la piste.

Pour rappel, la structure *si, sinon si, sinon* est de la forme suivante en Python :

```
if condition_1:
    action_1()
elif condition_2:
    action_2()
else:
    action_3()
```

Compter les pixels de différentes couleurs

Vous avez probablement constaté que prendre des décisions à partir d'un seul pixel n'étaient pas toujours le mieux. Par exemple il peut arriver que le pixel de la 5^{ème} ligne / 4^{ème} colonne soit blanc donc le robot décide d'avancer, alors qu'il y a un mur sur le côté et qu'il devrait plutôt tourner.

Pour améliorer le programme, nous allons considérer tous les pixels de l'image de la caméra. Tout d'abord nous allons compter combien de pixels sont noirs, rouges et blancs. Ensuite la décision sera prise ainsi : si la majorité des pixels sont rouge, tourner à droite ; si la majorité des pixels sont sombres, tourner à gauche ; et sinon aller tout droit.

Pour compter le nombre de pixels d'une certaine couleur, réalisez une suite de boucles *for* pour chaque composante de l'image : les lignes et les colonnes puis comptez pour chaque pixel quelle est sa couleur à l'aide des valeurs de rouge, de vert et de bleu qui la compose. Vous pouvez vous inspirer des seuils précédemment trouvés afin de déterminer la couleur du pixel. Vous pouvez commencer à partir du code suivant ou trouver votre propre façon de faire, il en existe plusieurs !


```
def count_pixel_colors(image: list):
# créer dict ici
    nombre_pxl_noir = 0
    nombre_pxl_rouge = 0
    nombre_pxl_blanc = 0
    for i in range(len(image)):
        ligne = image[i]
        for j in range(len(ligne)):
            pixel = ligne[j]
            canal_rouge = pixel[0]
            canal_vert = pixel[1]
            canal_bleu = pixel[2]
            if ...
                nombre_pxl_noir +=1
            ...
            else:
                ...
    return {"noir": nombre_pxl_noir, "rouge": nombre_pxl_rouge, "blanc":
nombre_pxl_blanc}
```

Astuce syntaxe Python pour les boucles *for*, au lieu d'utiliser des indices *i* et *j* pour accéder aux éléments d'un tableau, vous pouvez changer de syntaxe et écrire tout simplement :

for ligne in image:

 for pixel in ligne:

 rouge = pixel[0]

 vert = pixel[1]

 bleu = pixel[2]

Course de robots autonomes et poursuite des améliorations

A présent, vous pouvez commencer à organiser des courses entre vos robots.

Celles-ci seront encore plus pimentées si vous continuez de chercher à améliorer votre programme en modifiant par exemple : les vitesses de déplacement, les seuils des couleurs, des actions supplémentaires (en plus du tournant en pivotant sur place, un virage en avançant vers l'avant), etc.

Annexe : API de la librairie alphai

En programmation, la liste des fonctions accessibles d'un programme s'appelle API, pour *Application Programming Interface*, Interface de Programmation d'Application. Elle est utilisée par les programmeurs pour savoir comment interagir avec le programme. Vous trouverez l'API du module python **alphai** en suivant ce lien : [documentation de la bibliothèque alphai-api](#).

Vous l'aurez remarqué, le nom des fonctions est en anglais. Il est coutume dans le monde de la programmation d'écrire le code en anglais, pour qu'il puisse être compris par n'importe quel autre programmeur autour du monde, quelle que soit sa nationalité.