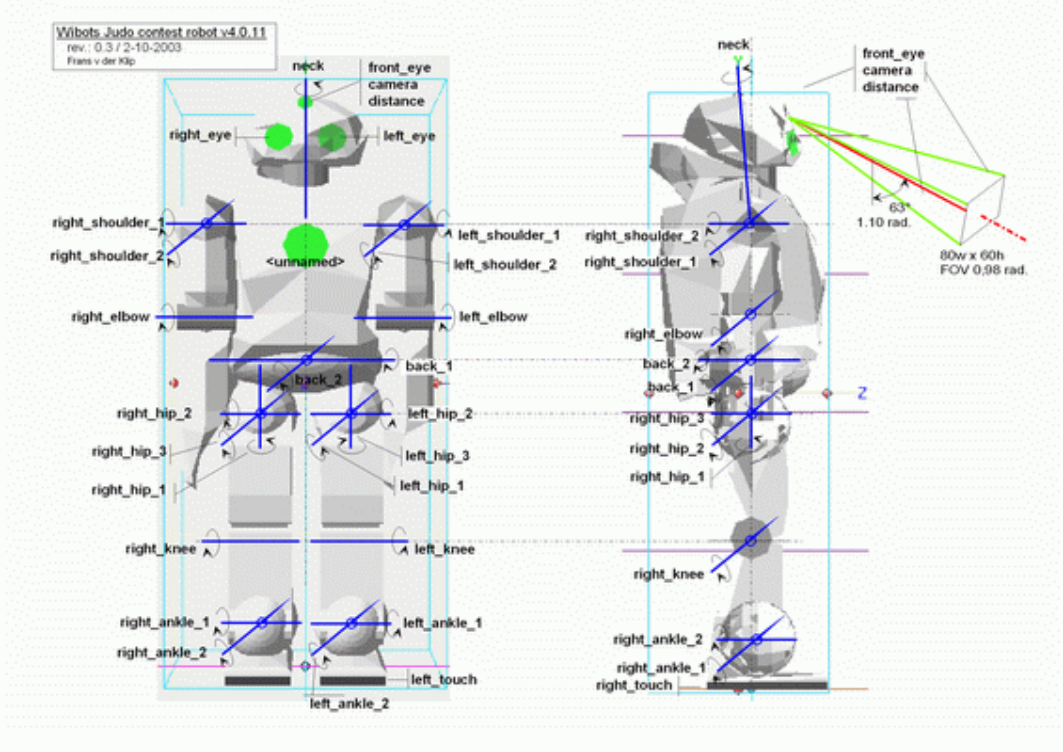
Devoir IA 2004

On se propose de développer une architecture logicielle agent pour contrôler un robot Judoka.



Pour cela, on utilisera le simulateur de robots en 3D Webots disponible au téléchargement ici :

http://www.cyberbotics.com/products/webots/download.html

Le robot se programme en Java et dispose de 22 moteurs, une caméra couleur embarquée, un capteur de distance dans la tête, deux capteurs de proximité dans les pieds, un inclinomètre et un compas. Il n'est pas nécessaire d'utiliser tous ces capteurs pour contrôler le robot.

Le choix de l'architecture logicielle controlant le robot est laissé aux choix des étudiants, mais il est conseillé de mettre en oeuvre l'ArchitectureDeSubsomption vue au ../TD6. L'objectif de ce devoir n'est pas de construire un robot combattant mais d'avoir un robot capable de se déplacer d'un emplacement à un autre du tatami et de se relever s'il tombe.

Optionel : Les étudiants qui le souhaitent pourront s'inscrire au concours proposé par Cyberbotics pour gagner un véritable robot avant le 7 mai 2004 : http://cyberboticspc1.epfl.ch/contest/ Une nouvelle compétition débutera bientôt sur le site http://roboka.org jusqu'au 2 mai 2005.

Un Wiki est disponible pour le concours : http://cyberboticspc1.epfl.ch/judowiki/ qui indique comment construire son controlleur en Java. Des fichiers exemples Java (judoka0.java et

judoka1.java) sont disponibles avec la version téléchargeable. L'API de contrôle du robot est ici : http://cyberboticspc1.epfl.ch/judowiki/More_20detailed_20Judoka_20API

Pour déplacer le robot plus simplement, on pourra débuter en utilisant le code disponible ici : http://cyberboticspc1.epfl.ch/judowiki/An_20interface_20that_20makes_20it_20more_20easy_20to_20compute_20movements

On pourra également se reporter à l'article : ARobustLayeredControlSystemForAMobileRobot pour un exemple complet d'utilisation d'une architecture de subsomption pour un robot explorateur.

Pour faire des expériences avec les moteurs, sélectionner un robot, puis choisir dans le menu Simulation, cliquer sur Show Robot Window. Les curseurs vous permettrons d'utiliser chacun des moteurs du robot.

Construire son controleur Java

- Insérer `import com.cyberbotics.webots.Controller;` en tête de votre classe,
- Faire dériver votre classe contrôleur de la classe `Controller`,
- Ajouter Controller.jar (qui se trouve dans le répertoire webots/lib) à votre variable d'environnement CLASSPATH,
- Les deux contrôleurs doivent porter le nom judoka0.java et judoka1.java. Vous pouvez réutiliser comme base de départ ces deux controleurs fournis dans le répertoire : webots/controlers/,
- Implémenter la méthode `reset` pour initialiser votre robot,
- Compiler votre contrôleur par `javac judoka*.java`.

Quelques éléments de l'API de Webots

- Méthode `void reset()` : initialise le robot à chaque début de round,
- Méthode `void die()` : appellé lors de la fin du round,
- Méthode `int robot_step(int time)` : attente du robot pendant time milliseconds,

Manipulation des devices

- Méthode `int robot_get_device(String device_name)` : retourne le numéro du dispositif matériel correspondant au nom. Chaque disposif matériel(device) dispose d'un numéro et d'un nom. Dans le cas du robot judoka, il y a :
  - 22 moteurs : back_1, back_2, left_hip_1, left_hip2, left_hip3, left_knee, left_ankle_1, left_ankle_2, left_shoulder_1, left_shoulder_2, left_elbow, neck, right_hip_1, right_hip_2, right_hip_3, right_knee, right_ankle_1, right_ankle_2, right_shoulder_1, right_shoulder_2, right_elbow, neck_tilt,
  - 1 caméra : camera,
  - un capteur de distance : distance,
  - deux capteurs de proximité : left touch et right touch,

    - une diode de couleur sur le front du robot : front eye,
    - un device correspondant au superviseur du robot : emitter. Le superviseur sert lors des phases d'apprentissages du robot. Par exemple, il est possible de communiquer avec le superviseur en lui envoyant des commandes comme : `stand me up X Y Z` pour remettre le robot droit à la position X, Y, Z, sans avoir besoin d'actionner les servo-moteurs.

Caméras

La caméra embarquée du robot a une taille de 80x60 pixels.

- Méthode `void camera_enable(int camera,int refresh)` : met en route la caméra spécifiée par le numéro avec une vitesse de rafraichissement en millisecondes,
- Méthode `void camera_disable(int camera)` : arrêt de la caméra correspondante,
- Méthode `int[] camera_get_image(int camera)` : récupère l'image courrante de la caméra spécifiée en paramètre sous forme d'un tableau d'entiers (pixel codé avec 8 bits au format RGB),

Servo-moteurs

Les moteurs sont en fait des servos-moteurs, c'est-à-dire qu'il est possible de connaitre leur position à chaque instant.

- Méthode `void servo_enable_position(int servo,int refresh)` : mis en route de l'échantillonage de la position du servo-moteur avec un rafraichissement avec une durée en millisecondes,
- Méthode `void servo_disable_position(int servo)` : arrêt de l'échantillonnage,
- Méthode `float servo_get_position(int servo)` : récupération de la position du servo-moteur,
- Méthode `void servo_set_position(int servo,float position )` : mettre le servo-moteur à la position indiqué,
- Méthode `void servo_set_velocity(int servo,float velocity)` : fixer la vélocité du servo-moteur correspondant,
- Méthode `void servo_set_force(int servo,float force)` : fixer la force du servo-moteur spécifié,
- Méthode `void servo_motor_off(int servo)` : arrêter le servo-moteur correspondant.

Capteurs de proximité

- Méthode `void touch_sensor_enable( int sensor,int refresh)` : active le capteur de proximité dont le numéro est passé en paramètre, les informations sont fournis avec une durée de rafraichissement en milliseconds,
- Méthode `int touch_sensor_get_value( int sensor )` : faire l'acquisition d'une valeur avec le capteur de proximité,
- Méthode `void touch_sensor_disable(int sensor)` : arrêt du capteur de proximité indiqué.

Superviseur du robot

- Méthode `void emitter_send(int emitter ,int size,byte[] message)` : envoyer une commande au superviseur.

À rendre

- Le code Java de votre robot (servira pour faire des matchs une fois le devoir rendu),
- Un rapport détaillant votre approche (format pdf) d'une dizaine de pages,
- Un fichier mpeg montrant votre robot en action.