

# Introduzione alla Simulazione Robotica

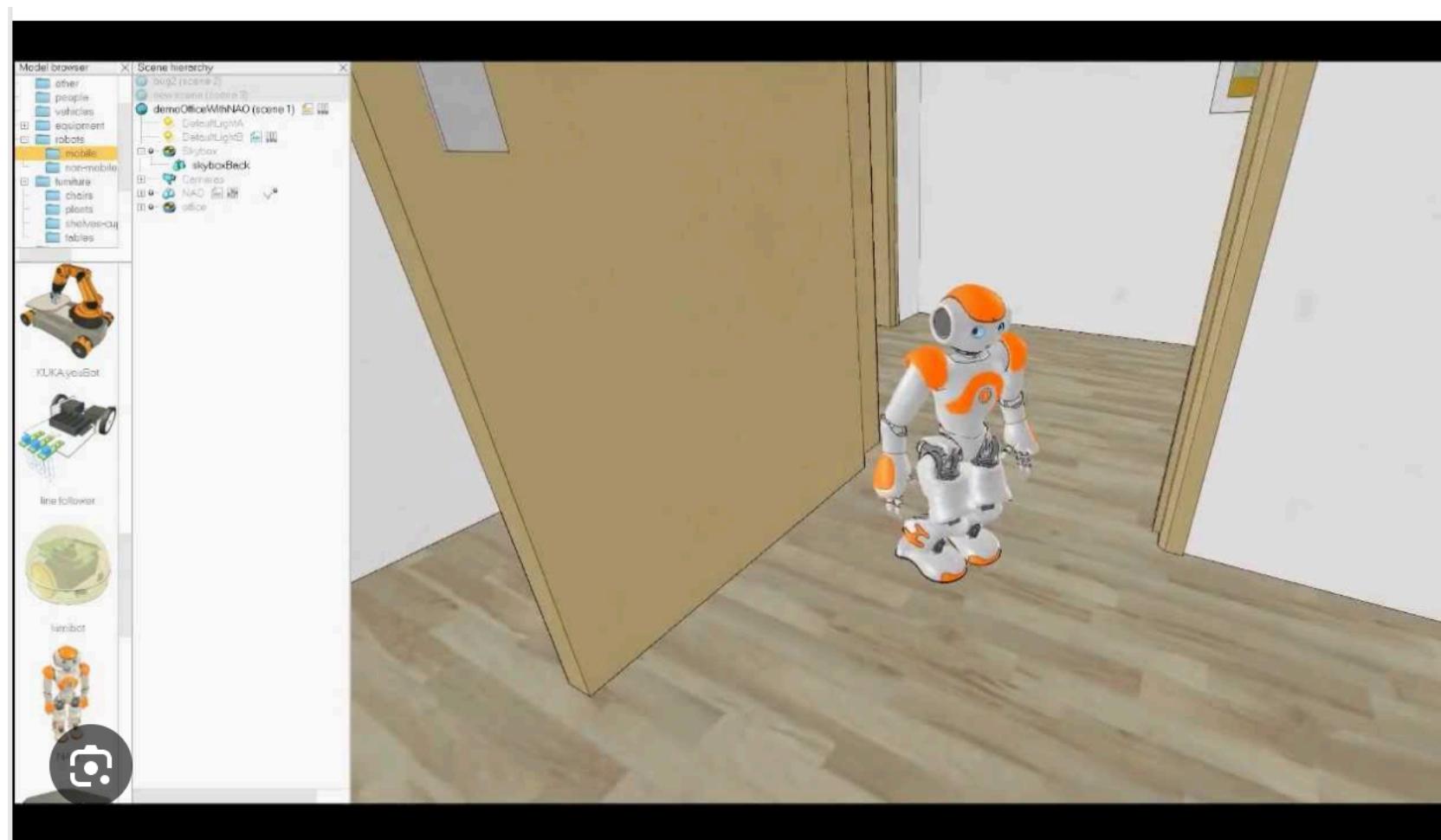
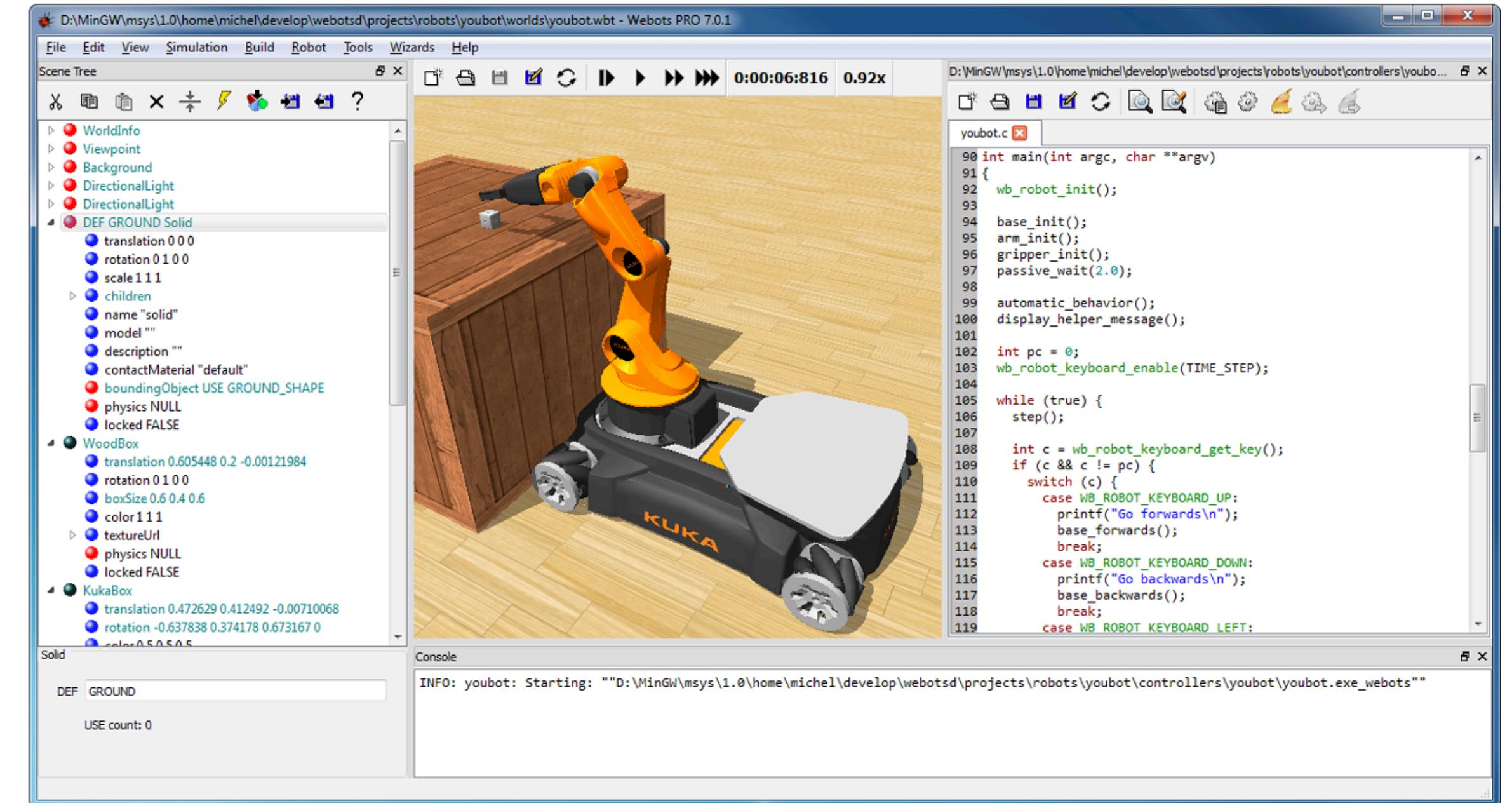
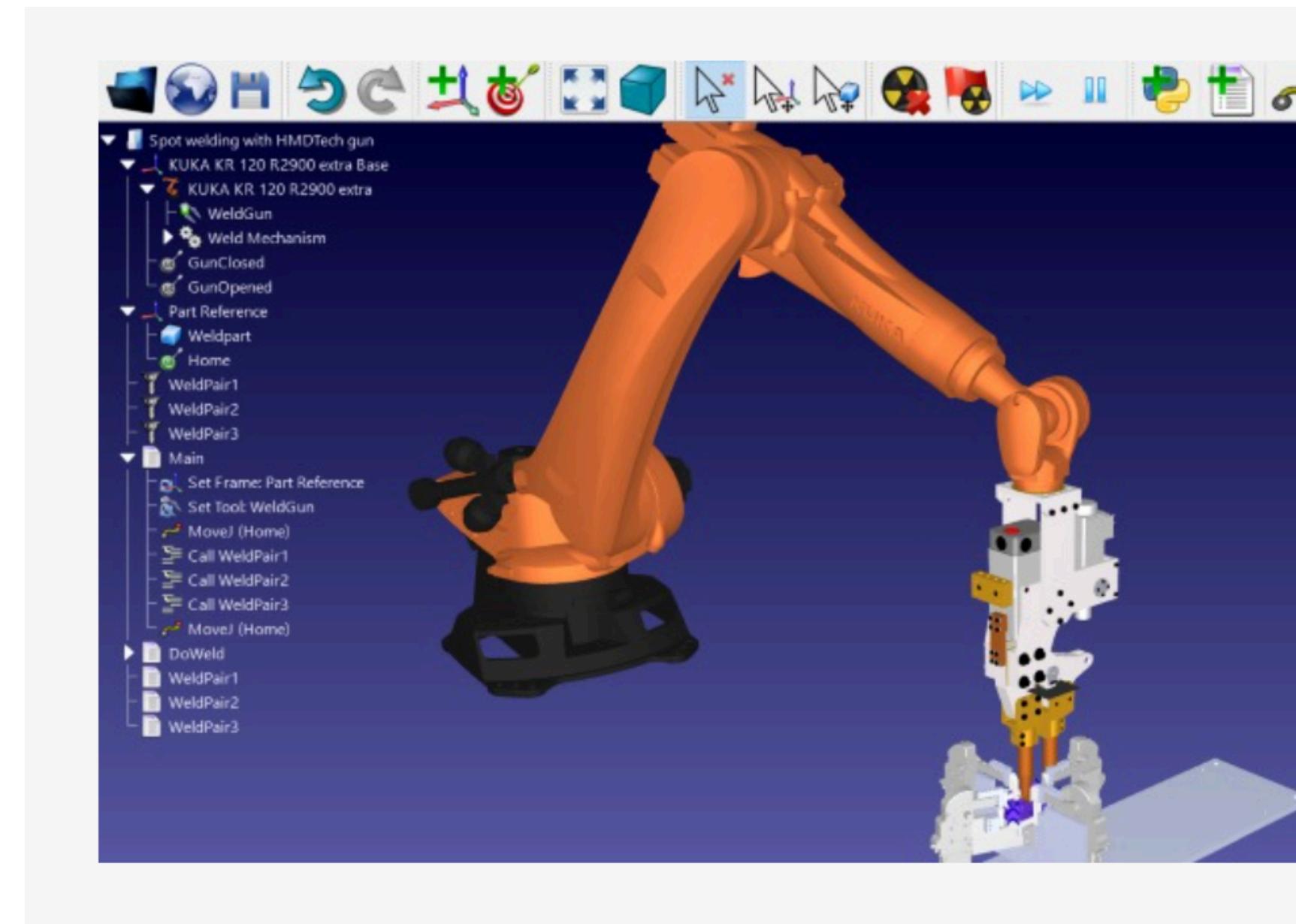
Webots

# Simulatore - definizione

- Strumento composto da uno o più elaboratori elettronici e da un apposito software capace di riprodurre il comportamento di un sistema governato da più variabili.

# A cosa serve un simulatore?

- Per **simulazione** si intende un modello della realtà che consente di valutare e prevedere lo svolgersi dinamico di una serie di eventi o processi susseguenti all'imposizione di certe condizioni da parte dell'analista o dell'utente.
- Un simulatore di volo, ad esempio, consente di prevedere il comportamento dell'aeromobile a fronte delle sue caratteristiche e dei comandi del pilota.



# A cosa serve un simulatore?

- Ambiti scientifici e tecnologici
  - impossibilità di riprodurre fisicamente in laboratorio una situazione reale da studiare
  - abbiamo bisogno di una trasposizione in termini logico-matematica-procedurali di un "*modello concettuale*" della realtà;

# Usi dell simulazione

- Analizzare un fenomeno e studiarne l'evoluzione prima della sua effettiva implementazione
- Apprendimento, esercitazioni, gioco
- Testing - logica inversa della simulazione

# Elementi caratteristici di un modello di simulazione

- Entità - Le entità sono gli elementi "trattati" dal processo di simulazione, appartengono ad un environment e rispettano i principi di fisica del dominio del problema sotto studio
- Operazione: rappresenta una delle trasformazioni che avranno luogo sull'entità.
- Stati: gli stati sono delle variabili (di tipo vario: possono essere numeri o valori logici) che descrivono lo stato del sistema e delle sue componenti, per ogni istante di tempo.
- Eventi: fenomeni che modificano lo stato del sistema (ad esempio, la fine di una lavorazione modifica lo stato di una macchina da "occupata" a "libera").
- Code: insiemi di entità che non possono accedere alle trasformazioni successive in quanto la macchina risulta occupata.
- Attributi: proprietà permanenti di un insieme di entità o di una macchina.
- Orologio locale: orologio che contiene, a livello di singola macchina, l'istante di tempo che identifica la fine della lavorazione in corso.
- Orologio generale: orologio che regola lo scorrere generale del tempo di simulazione.

# Simulazione

- Il processo di simulazione prevede una fase di modellazione, una di esecuzione ed una di analisi dei risultati
- Al di sotto di un qualunque processo di simulazione c'è una **architettura di controllo**
  - il modello sul quale organizzare il sistema di controllo del robot nel simulatore

# Simulazione e controllo

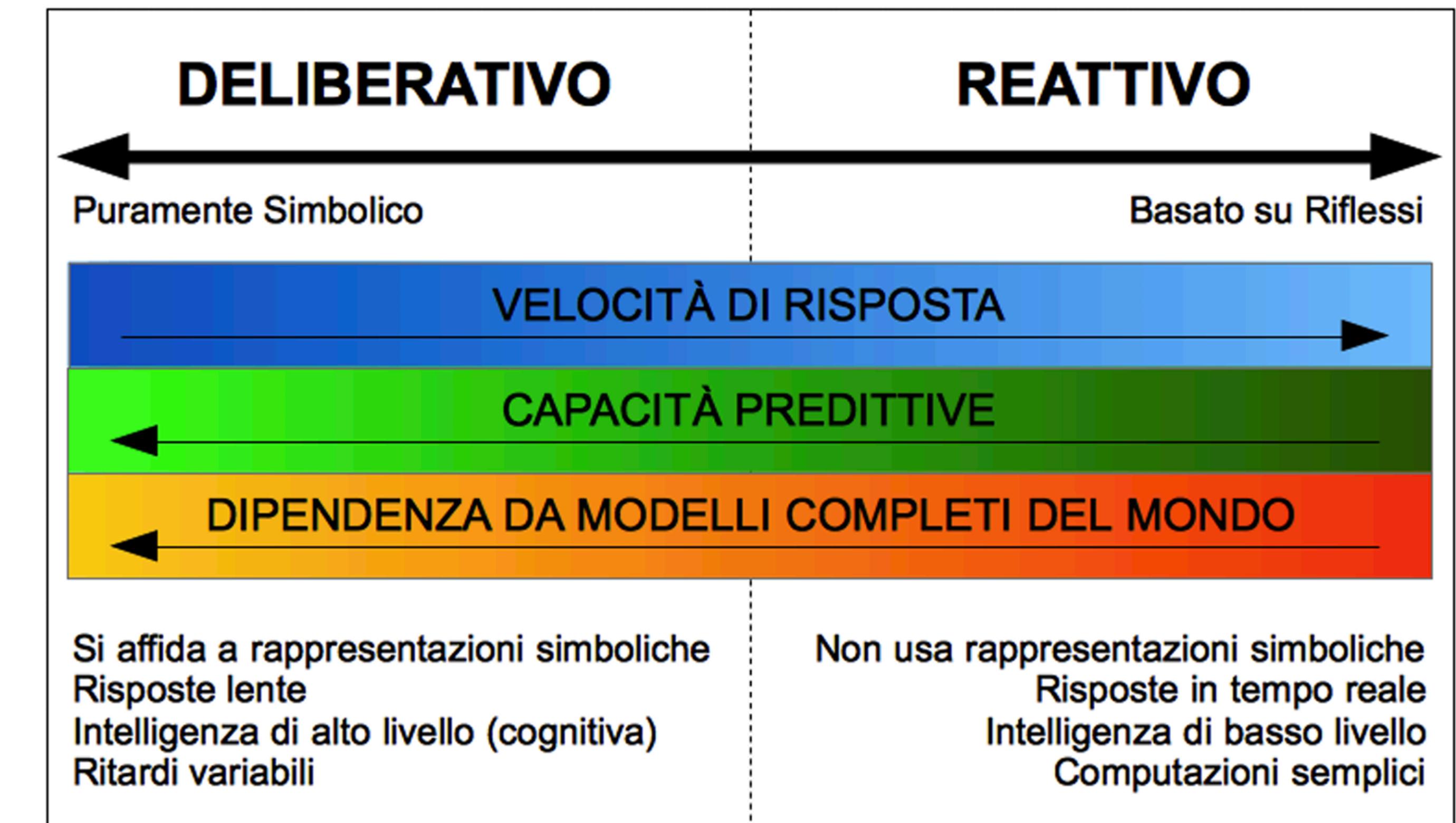
- Classificazione principale, controllo:
  - deliberativo
  - reattivo
- Autonomia

# Simulazione e controllo

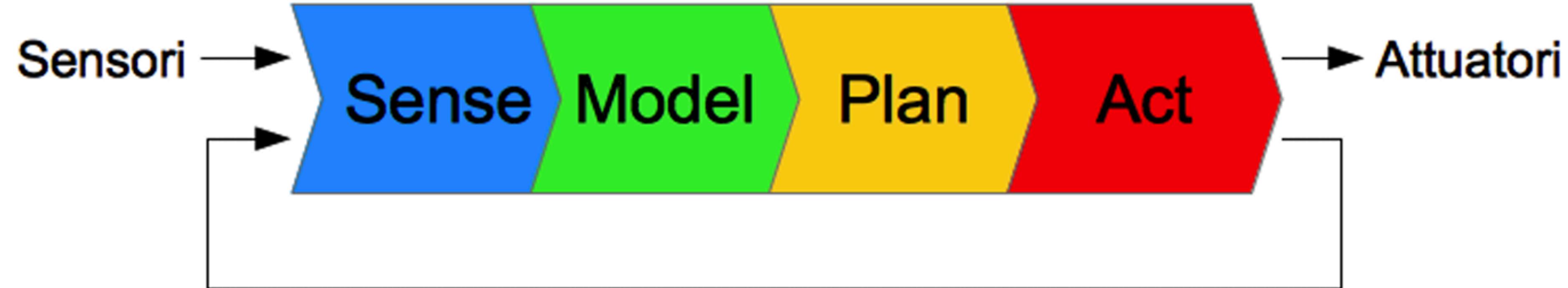
- Classificazione principale, controllo:
  - deliberativo
  - reattivo
- Autonomia

# Simulazione e controllo

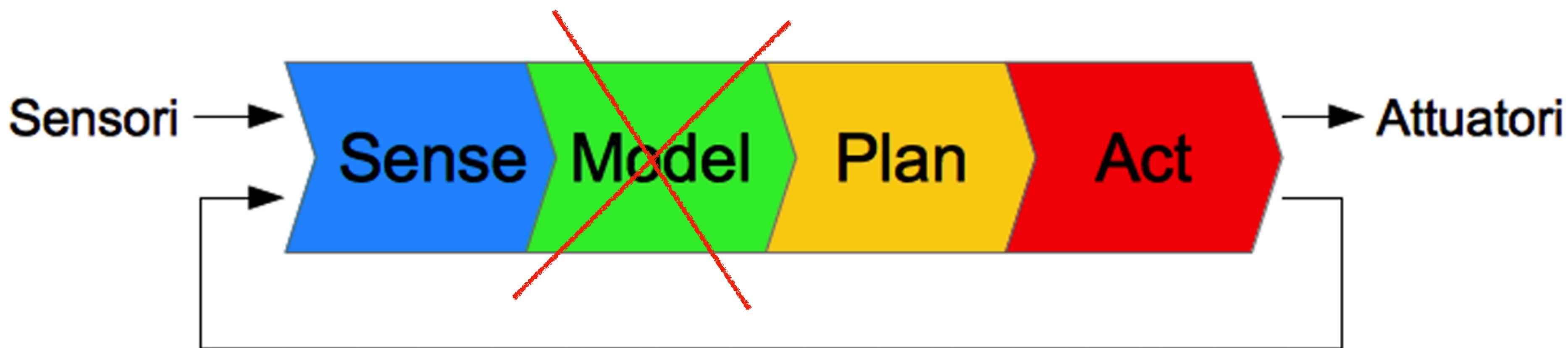
- Reattività all'ambiente
- Comportamento intelligente
- Integrazione di sensori multip



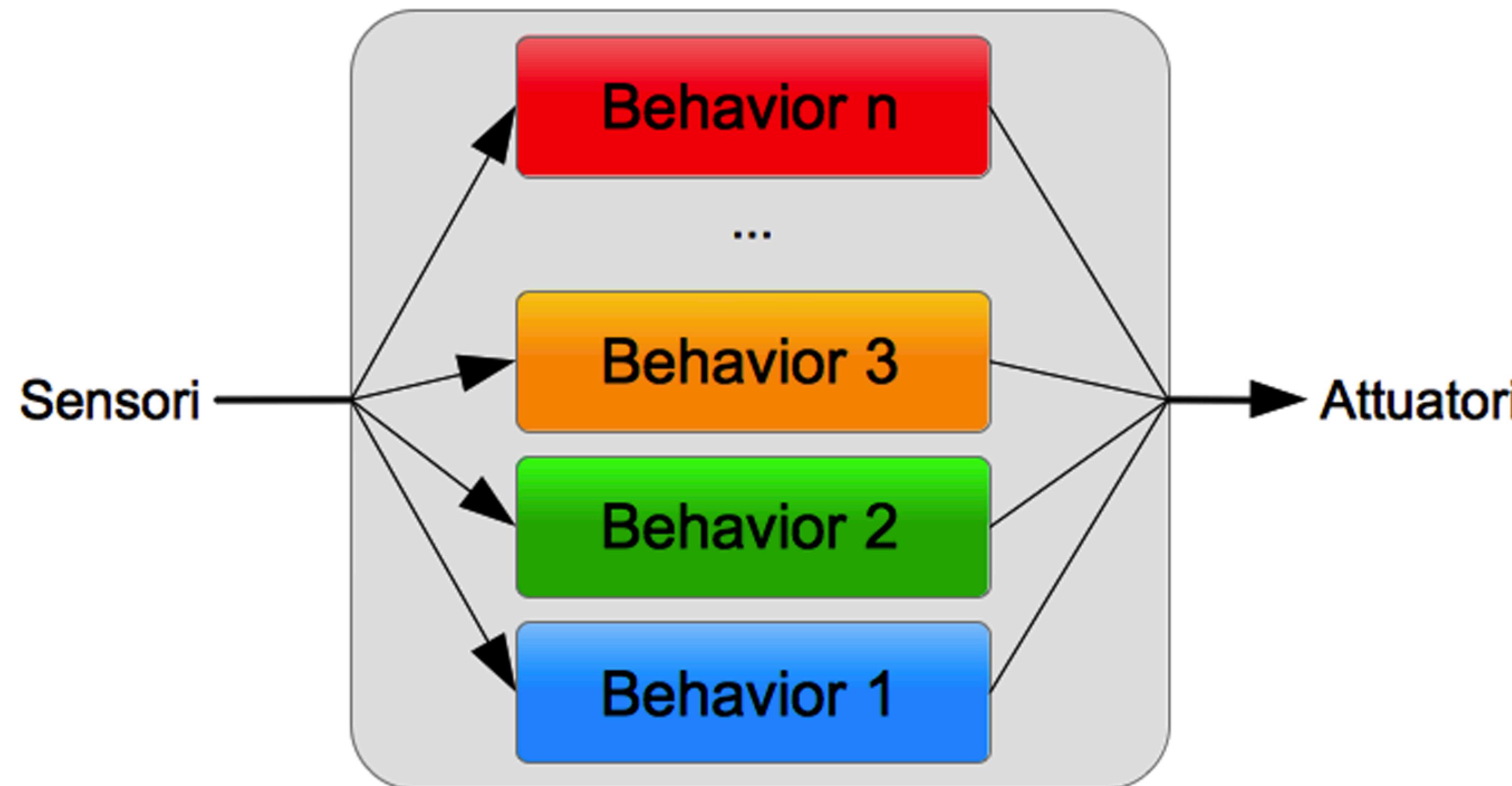
# Controllo Deliberativo



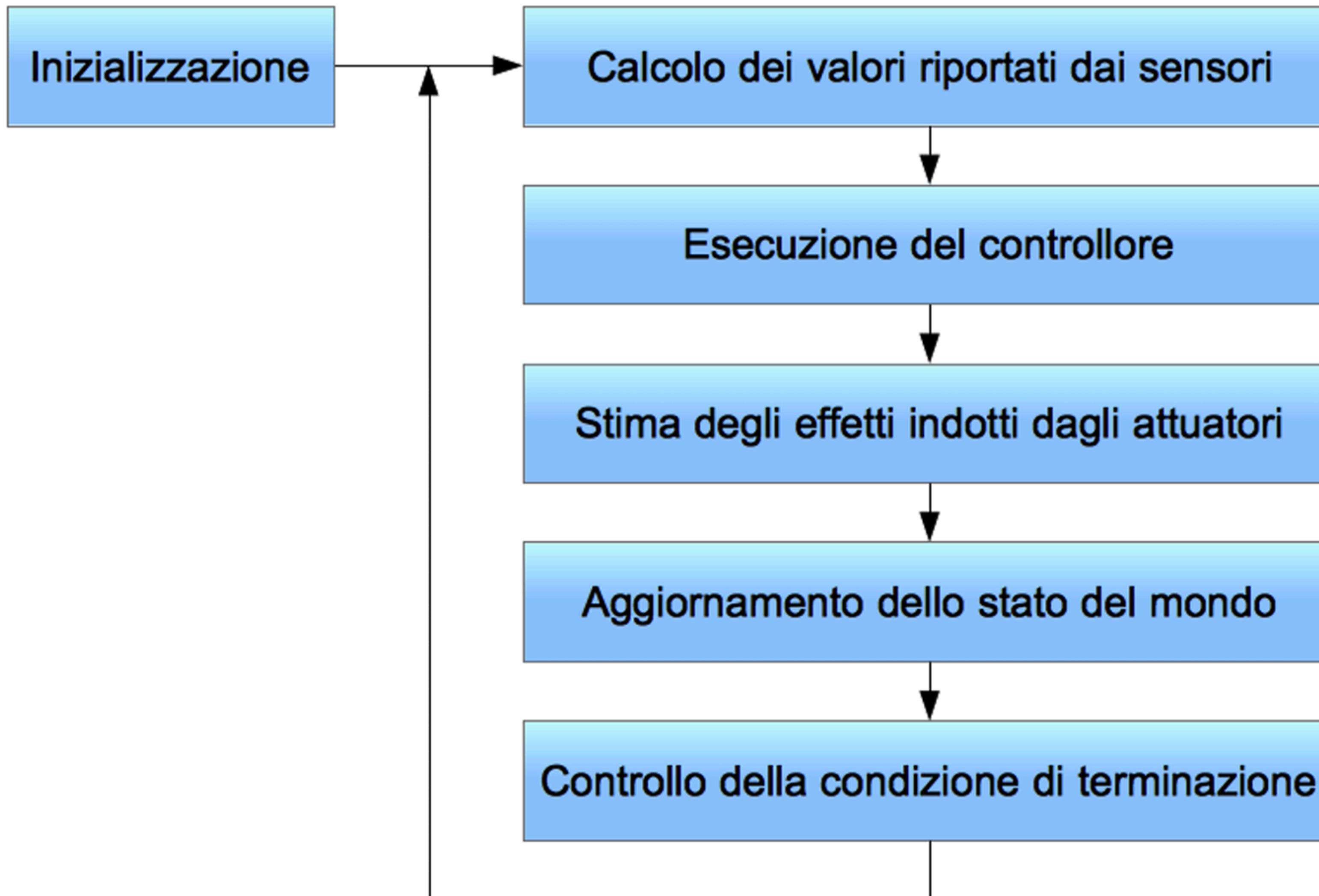
# Controllo Reattivo



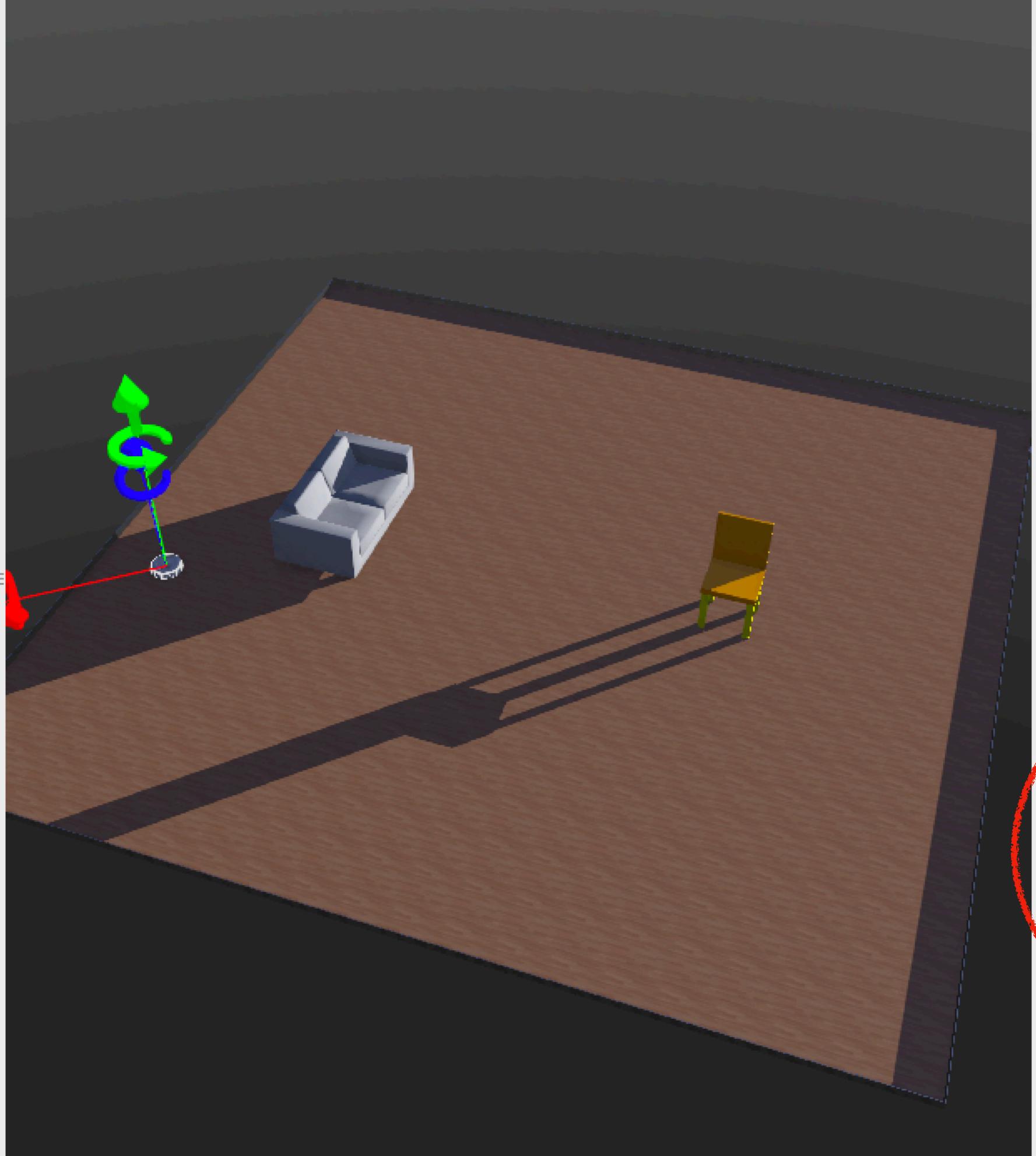
# Controllo Behaviour-Based



# Flusso esecutivo di una simulazione



Simulation View



0:03:02:400 - 0.00x

- translation 0 0 0
- rotation 0 1 0 0
- name "rectangle arena"
- contactMaterial "default"
- floorSize 10 10
- floorTileSize 0.5 0.5
- > floorAppearance Parquetry
- wallThickness 0.01
- wallHeight 0.1
- > wallAppearance BrushedAluminium
- > Create "Create"
- translation -4.19 0.0442 1.35
- rotation -0.000111-0.000755 -2.4
- name "Create"
- controller "create\_avoid\_obstacles"
- controllerArgs
- customData ""
- supervisor FALSE
- synchronization TRUE
- bodySlot
- > Sofa "sofa"
- > SimpleChair "simple chair"

Selection: controller (String)

create\_avoid\_obstacles

Select... Edit

...ations/Webots.app/projects/robots/irobot/create/controllers/create\_avoid\_obstacles/create\_avoid\_obstacles.c

x create\_avoid\_obstacles.c

```

187 do {
188     double l = wb_position_sensor_get_value(left_position_sensor) - l_offset;
189     double r = wb_position_sensor_get_value(right_position_sensor) - r_offset;
190     double dl = l * WHEEL_RADIUS; // distance covered by left wheel in
191     double dr = r * WHEEL_RADIUS; // distance covered by right wheel in
192     orientation = neg * (dl - dr) / AXLE_LENGTH; // delta orientation in radian
193     step();
194 } while (orientation < neg * angle);
195 stop();
196 step();
197 }

/* main */
199 int main(int argc, char **argv) {
200     wb_robot_init();

203     printf("Default controller of the iRobot Create robot started...\n");

205     init_devices();
206     srand(time(NULL));

208     wb_led_set(leds[LED_ON], true);
209     passive_wait(0.5);

211     while (true) {
212         if (is_there_a_virtual_wall()) {
213             printf("Virtual wall detected\n");
214             turn(M_PI);
215         } else if (is_there_a_collision_at_left() || is_there_a_cliff_at_left()) {
216             printf("Left obstacle detected\n");
217             go_backward();
218             passive_wait(0.5);
219             turn(M_PI * randdouble());
220         } else if (is_there_a_collision_at_right() || is_there_a_cliff_at_right() || is_there_a_cliff_at_right()) {
221             printf("Right obstacle detected\n");
222             go_backward();
223             passive_wait(0.5);
224             turn(-M_PI * randdouble());
225         } else {
226             go_forward();
227             fflush_ir_receiver();
228             step();
229         }
230     }
231     return EXIT_SUCCESS;
232 }

```

# Webots

- Pacchetto software professionale per la simulazione di robot mobili.
- Offre un ambiente di prototipazione rapida che consente all'utente di creare mondi virtuali 3D con proprietà fisiche quali massa, giunti, coefficienti di attrito, ecc. L'utente può aggiungere semplici oggetti passivi o oggetti attivi chiamati robot mobili.
- I robot possono avere diversi schemi di locomozione (robot a ruote, robot a gambe o robot volanti).
- Possono essere dotati di una serie di sensori e attuatori
  - sensori di distanza, ruote motrici, telecamere, motori, sensori tattili, emettitori, ricevitori, ecc.
- L'utente può programmare ogni robot individualmente per esibire il comportamento desiderato.
- Webots contiene un gran numero di modelli di robot ed esempi di programmi di controllo per aiutare gli utenti a iniziare.
- Webots contiene anche una serie di interfacce con robot mobili reali, in modo che, una volta che il robot simulato si comporta come previsto, sia possibile trasferire il suo programma di controllo a un robot reale come e-puck, DARwIn-OP, Nao, ecc. L'aggiunta di nuove interfacce è possibile attraverso il relativo sistema.

# La simulazione con Webots

- Una simulazione Webots è composta dai seguenti elementi:
  - Un file mondo Webots (.wbt) che definisce uno o più robot e il loro ambiente. Il file .wbt a volte dipende da file PROTO (.proto) e texture esterni.
  - Uno o più programmi di controllo per i suddetti robot (in C/C++/Java/Python/MATLAB).
  - Un plugin di fisica opzionale che può essere usato per modificare il comportamento fisico regolare di Webots (in C/C++).

# Elementi principali

- World - è una descrizione 3D delle proprietà dei robot e del loro ambiente. Contiene una descrizione di ogni oggetto: posizione, orientamento, geometria, aspetto (come colore o luminosità), proprietà fisiche, tipo di oggetto, ecc. I mondi sono organizzati come strutture gerarchiche in cui gli oggetti possono contenere altri oggetti (come in VRML97).
- Controller - è un programma per computer che controlla un robot specificato in un file mondo. I controllori possono essere scritti in uno qualsiasi dei linguaggi di programmazione supportati da Webots: C, C++, Java, Python o MATLAB.
- Supervisor controller - può eseguire operazioni che normalmente possono essere eseguite solo da un operatore umano e non da un vero e proprio robot. Il Supervisor ha accesso ad operazioni privilegiate. Le operazioni privilegiate comprendono il controllo della simulazione, ad esempio lo spostamento dei robot in una posizione casuale, la cattura video della simulazione, ecc.

# Filosofia di sviluppo di Webot



**2 program**

```
#include <webots/robot.h>
#include <webots/distance_sensor.h>

int main(int argc, char *argv[]) {
    wb_robot_init();
    WbDeviceTag camera = wb_robot_get_device(WB_CAMERA_NAME);
    wb_camera_enable(camera, TIME_STEP*1000);

    for (int i = 0; i < 8; i++) {
        char device_name[10];
        sprintf(device_name, "camera%d", i);
        wb_camera_set_name(camera, device_name);
    }
}
```

