# MOBILE ROBOTICS
# follow-up
# EXPLAINABLE AI - part 1

Serena De Antoni
Alberto Righetti

Mobile Robotics and Explanaible AI courses
Master’s degree in Artificial Intelligence

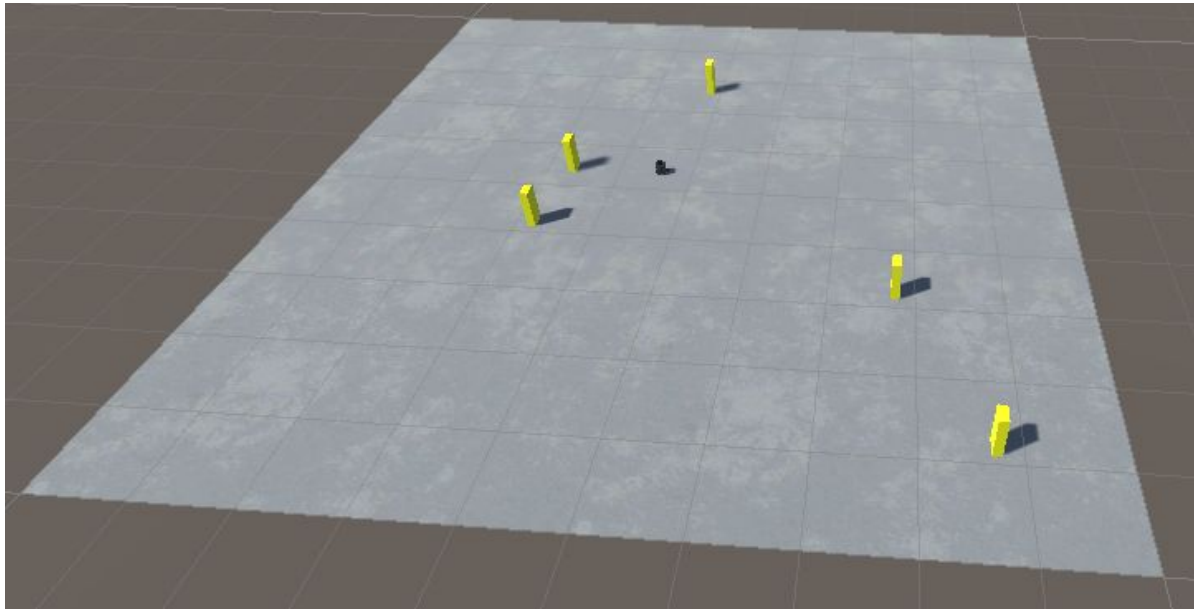
# Goal

A solution of the **RockSample** problem:
a python implementation of a **ROS** node that uses the **Clingo API** to send the initial conditions in ASP format, which calculates a plan of action considering as **good only the rocks at a distance less than a given threshold**. The robot reaches the goal matching each action with a motion primitive and moving with the actions of the plan in sequence, being careful to not collide with the rocks.

# Environment

The entire testing process was conducted first in a simulated Unity environment and then in a real environment:

- 10 x 10 arena
- 5 rocks
  - 4 good
  - 1 not good



# Code Consideration

Only one ROS node

3 main phases:

**lidar phase:**
- obtain the objects distances
- calculate the discrete position

**clingo phase:**
- add additional constraint to the control program of rocksample.lp
- return the action plan

**motion phase:**
- rotation
- movement
- sample
- collision

# Lidar Phase

1. Extract from the **lidar signal** the presence of objects within a certain threshold.
   - We return the **distances** and the **angles** between the turtlebot and the rocks.

2. Given the distances and the angles:
   - compute relative **x and y distance from the agent**
   - compute the **position discrete** of each obstacle in the grid arena

# Clingo Phase: ASP

Adding to the ASP program:

- position of agent
- positions of rocks
- rocks distances
- threshold in which obstacles have to be sampled

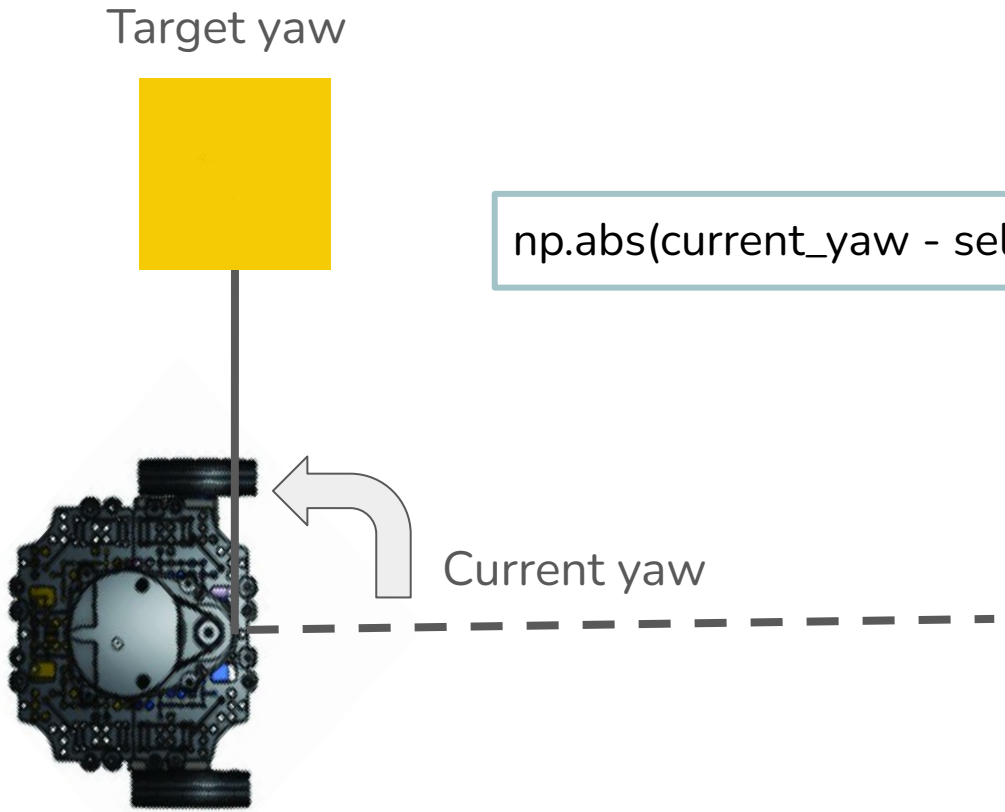
return the **action plan**

# Motion Phase

Given the action plan the turtlebot, using the odometry, has to :

- **rotate** to align with the action (N, S, E, W)
- **move forward** for a certain distance(step)
- do the **sample** if the plan action requires it
- avoid **collision**

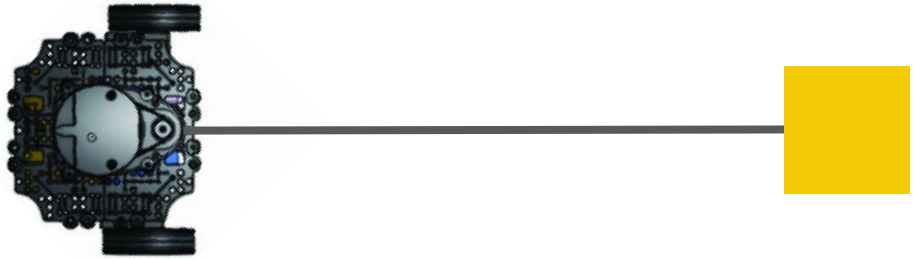
# Code Consideration : Rotation




```
np.abs(current_yaw - self.target_orientation) >= 10
```

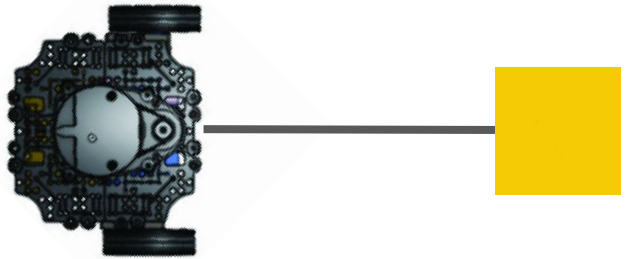
# Code Consideration : Step

p1 = Start position



np.abs(p2 - p1) < self.step_foreward

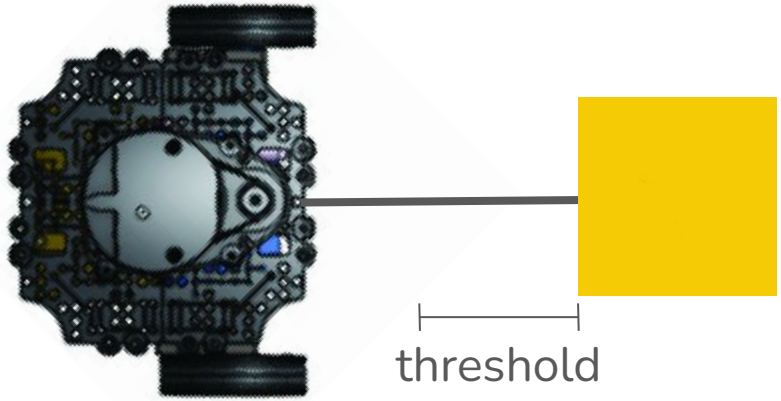
p2 = Current position



# Code Consideration : Collision

```
np.nanmin(np.concatenate([ranges[pos_positive],ranges[pos_negative]])) <
                                                    self.threshold_collision
```

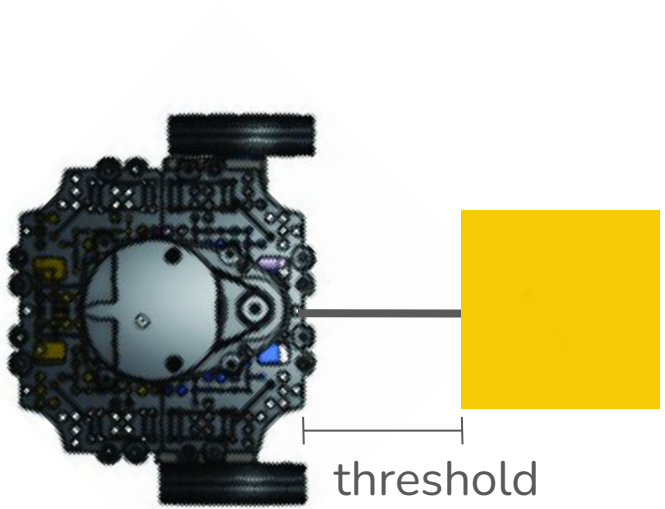
**stop** the robot
and
start with the **new action**



# Code Consideration : Sample

```
self.action == 'sample'
```

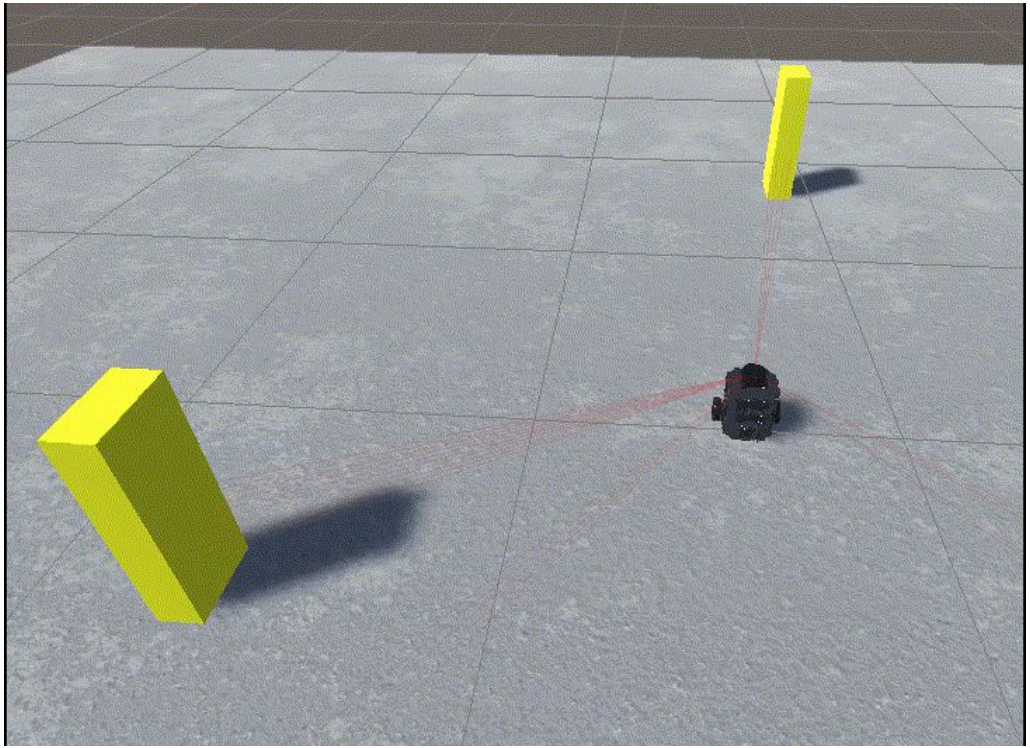
**stop** the robot
and
start with the **new action**



# Code Consideration : Some assumption

- the robot in the initial position is oriented towards the north
- the robot is located in the center of the cell
- walls:
  - in Unity simulation there are no walls
  - in real simulation there is a threshold for the lidar phase

# Results



# Thanks for your attention