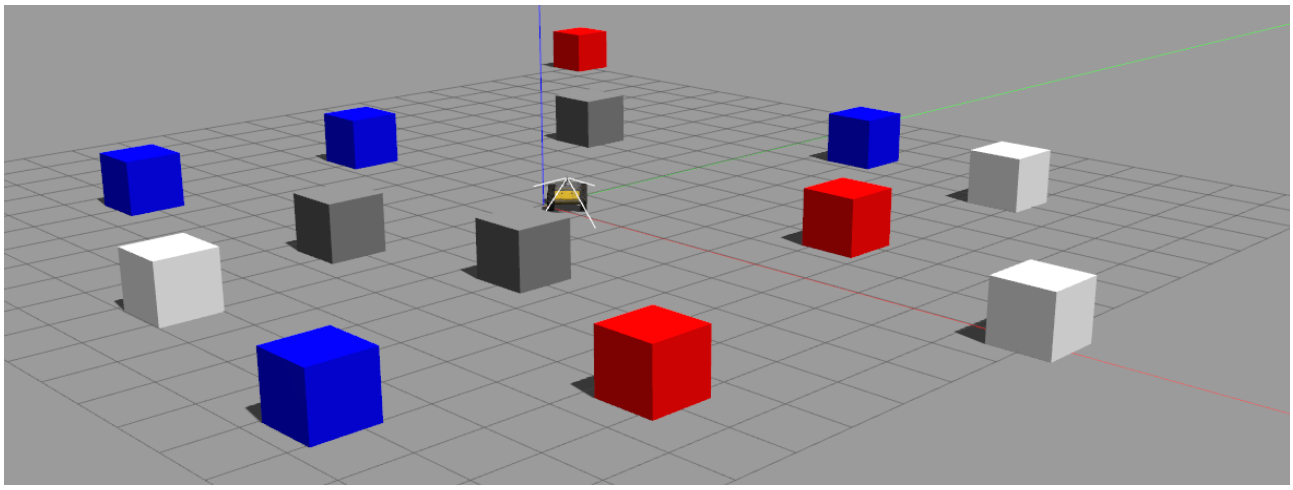
Coding task

description:

The starting point (what is given to you) is a simulated gazebo world with a robot and multiple boxes of different colors.



The goal of the task is to implement, in a ros node, one of the two behaviors described bellow:

a) at start time, the robot must request its next goal to a ros service */get_point*. It receives a 3d point to which the robot must go (ignoring last coordinate) avoiding collisions with the boxes. The point may lay inside of a box. This must be inferred and solved (eg. by replacing the point with another one outside the box). After the robot reaches the point it will request another goal.

b) at start time, the robot must request its next goal to a ros service */get_color*. It receives a string containing a color name. The robot must turn around detecting boxes of the given color. It must choose one, estimate its pose, navigate towards it, stop nearby and request the next goal.

preparatory steps:

- Install ROS in your machine
- Clone the provided catkin workspace at https://github.c/EloyRC/catkin_hbp_task and build it running catkin_make from the root folder (it has been tested in Ubuntu 20.04 / ROS noetic but it should work in melodic or even kinetic)
- Add these lines to your .bashrc file and source it:

```
source /opt/ros/noetic/setup.bash
export CATKIN_WS=/replace/with/the/path/to/catkin_hbp
source $CATKIN_WS/devel/setup.bash
export GAZEBO_MODEL_PATH=$CATKIN_WS/src/gazebo_environment/models
```

- You can check that everything is ok by running:

```
roslaunch gazebo_environment hbp_husky.launch
```

gazebo should start correctly and running and rosservice list output should include *get_color* and *get_point.*

task instructions:

The definitions of the involved ROS services are stored in *gazebo_msgs2* folder. You can look there for details.

The robot model is contained in file *gazebo_environment/models/husky_model/model.sdf.* It has a camera mounted, you can mount additional sensors if required.

The robot pose is provided via the topic */odom* of type *nav_msgs/Odometry.*

The robot speed is controlled by publishing to topic */husky/cmd_vel* of type *geometry_msgs/Twist*.

Implement any of the two behaviors described above in a ros node.

Assessment criteria:

Completing the task is of course desirable but more important is clarity and quality of code, showing, when possible, good design principle and knowledge about the chosen programming language (you can choose between C++ or python according to your preference).

Submitting the task:

you can implement your solution in a separate branch or fork of the repository and, after completion, open a pull request, which I will receive. You can additionally send an email to me.