

**DISIM, University of L’Aquila**

**Robot Maze Search**

Intelligent Systems and Robotics Laboratory, prof. Giovanni De Gasperis

**By Yuna Frolov**

**Homework Specifications:**

1. Read the Homework 2 document in the previous section
2. Write Python code with classes as requested
3. Embed this code is your brain think method
4. Produce a screencast video of the running simulation (.ogv or .webm format), with Python console output and V-REP world both visible
5. Zip the V-REP scene and Python Project in a single ZIP file
6. Write an accompanying documentation file

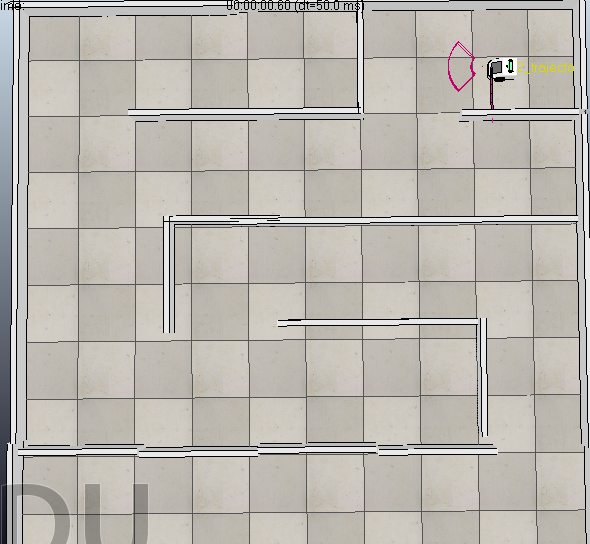
There are 3 files to deliver:

1. documentation
2. zip of the python code
3. video

**Preconditions:**

* 1. Have the files: *remoteApi.dll, vrep.py, vrepConst.py* in the same folder as project
  2. Have **vrep** and **numpy** libraries installed on the machine

Since my previous homework scene was not suitable for a maze search (it was an office scene), I have made alterations to the environment.



**V-Rep file:**

|  |  |
| --- | --- |
| File name | RobotMazeSearch.ttt |

Description: In the scene there is a robot of the type **dr12** *(instead of the pioneer I had in the previous homework, since I was working with only two sensors as opposed to the 16 I had before),* with 2 proximity sensors I added – one is a cone and one is a ray. Inside its LUA code there is only a connection to the API server.

I have created a maze for the robot to search for a goal position in, starting point is at the north and the goal is at the opposite side. The robot must search through the maze for an exit while avoiding hitting the walls.

**Python files:**

|  |  |
| --- | --- |
| File name | robotControllMain.py |
| Purpose | This file created the objects of the World and the Robot brain, attempts the connection to the simulation and actuates the code by making the robot “think” |
| Functions | Main function: creating objects, calling the robots’ “think” function |

|  |  |
| --- | --- |
| File name | \_\_init\_\_.py (in RobotWorld folder) |
| Purpose | All of its functions allow it to communicate with the simulation and lead the robot to move around the maze while avoiding the walls. The robot uses “wall follower” algorithm to follow the left wall until the goal position is found. The idea is from a solution to a maze in which a person is touching the wall of a maze without letting go, thus finally reaching the exit. |
| Classes | Class World  Class RobotBrain  Class RobotState  Class MazeWorld  Class Heuristic  Class RobotHeuristic |

|  |  |
| --- | --- |
| Class | Class World |
| Functions | * **\_\_init\_\_** : this is the initialization function, and it starts the simulation through connection to the given ip and port. If the connection was successful – the initialization is complete. If not – a message appears and the application is stopped. * **Sense :** This function collects the data from the robot’s sensors * **Act:** This function receives a velocity for the left and right sensors and communicates the new values to the robot simulation. * **Close:** this function closes the connection and ends the simulation. |

|  |  |
| --- | --- |
| Class | Class RobotBrain |
| Functions | * **\_\_init\_\_:** initialization of self. * **Perception:** Creates a representation of the maze world and state of robot. * **Decision:** The robot decides how to act according to the action number he receives from think function. * **Think:** The robot thinks by perceiving his surroundings. He sees the front sensor and the left sensor and decides: if the front sees no wall, and the left one is touching a wall – go straight. If there is a wall in front – turn right. If you lose the left wall – try going left to find it. |

|  |  |
| --- | --- |
| Class | Class RobotState |
| Functions | * **\_\_init\_\_:** initialization of self. * **getH:** returns the heuristic of the state |

|  |  |
| --- | --- |
| Class | Class MazeWorld |
| Functions | * **\_\_init\_\_:** initialization of self. * getState: returns the current state of the robot. * solution: if the robot got to the goal – return true, thus making it stop and simulation is killed. |

|  |  |
| --- | --- |
| Class | Class Heuristic |
| Functions | * **\_\_init\_\_:** initialization of self. * **H:** base for heuristic function |

|  |  |
| --- | --- |
| Class | Class RobotHeuristic |
| Functions | * **H:** the heuristic function: “wall follower” algorithm: follow the left wall until the goal position is found. The idea is from a solution to a maze in which a person is touching the wall of a maze without letting go, thus finally reaching the exit. |