Practical Exercise 2: Bug Algorithm

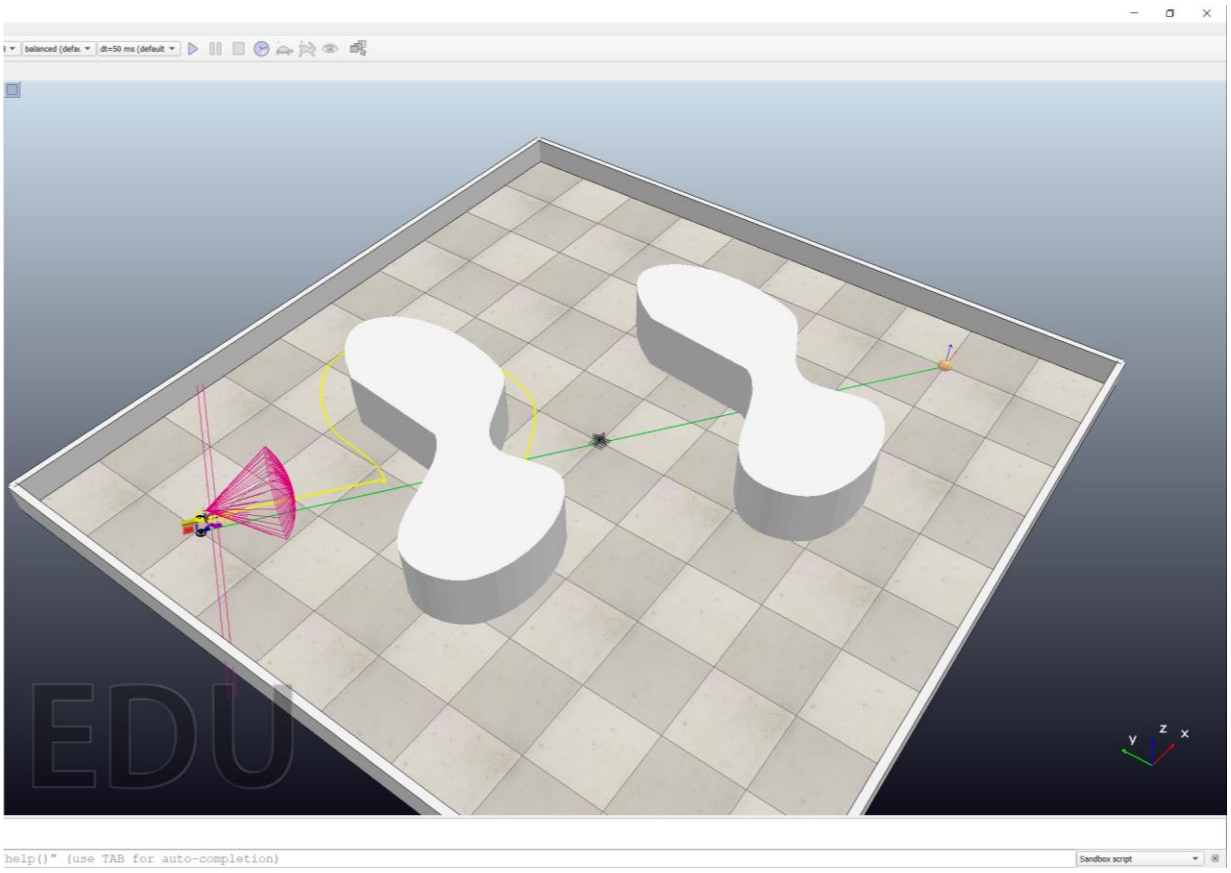
Objective:

In this practical exercise you implement a motion planning algorithm for a robot with a differential drive. This robot has only local knowledge about its environment. The robot has a proximity sensor in the front and two ray sensors on each side. The BUG 2 algorithm should be implemented.

For this purpose you implement in Python (here we provide a source code framework) or C++ (here, we **do not provide** a source code framework) and evaluate your results by visual inspection using CoppeliaSim.

Overall Task:

In the scene *dyor_bug.ttt* you see this robot, which should move from the start to the goal position. This way is obstructed by two obstacles (see figure).



You should implement a motion planning algorithm, where you to steer the robot in the direction to the goal, while avoiding obstacles using the BUG 2 algorithm.

Task 1:

Move the obstacles out of the way of the robot and implement the motion to the goal phase.

Task 2:

Implement moving the robot around the obstacle. It is sufficient moving around the obstacle(s), until the robot hits the m-line

Watch the video

https://www.youtube.com/watch?v=r03dFeZA2SY&list=PLjzuoBhdtaXOoqkJUqhYQletLLnJP8vjZ&index=44

to learn how to control the robot for that purpose.

Task 3:

Implement BUG 2 moving the robot around the obstacles. Use a state machine for that purpose. This is partlyimplemented in the sample code, where you have also to implement the functions:

```
def action_moving(self):  # step motion to goal

def action_rotating(self): # step rotation tot he desired orientation

def action_rounding(self): # step wall following
```

Preparation:

- Make yourself familiar with the given code example
- Make yourself familiar with the BUG 2 algorithm
- Have a scaffold of your program ready

Report:

The report should contain
- the task description itself (you can reuse this document, at the top your name(s) must be mentioned)
- a comprehensive description of the implemented algorithm
- Sketches and explaining figures
- source code with result output
- you deliver one zipped file. It contains the report as PDF, the .ttt scene files and the solution python files including all additional project files necessary, to run the program. Your program must start out of the box.
- You can provide one solution for one group consisting of max. 2 students.

Naming Conventions: Practical_Exercise_2_Student1_Student2.zip