Projects

Challenge levels and grades

Projects can be completed at three *Challenge levels*. The *Challenge level* determines the **best** grade that can be received to the project!

Challenge level	Best grade
Basic	3
Advanced	4
Epic	5

Tip

The projects are defined in a way that it is recommended to tart with the **Basic** level, and then gradually work towards **Epic**.

The projects are graded based on the follwoing aspects:

- Proved to be the student's own work
- Running results valid output
- Usage of versioning, usage of GitHub/GitLab/other repository
- Grading:
    - completeness of the soultion
    - proper ROS communication
    - proper structure of the program
    - quality of implementation
    - documentation quality

Schedule

Week	Date	Event
2.	szept. 13	Announcement of project topics.
8.	okt. 25	Project milestone.
14.	nov. 6	Project presentations.

Grading

Personal attendance on the classes is mandatory (min 70%).

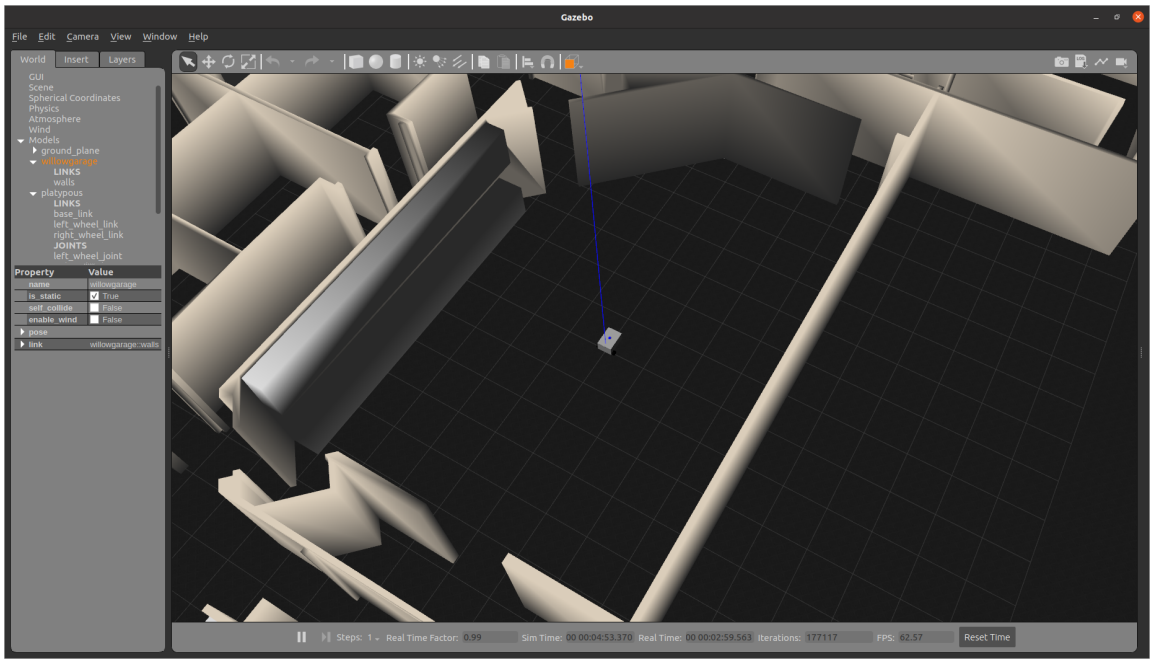
To pass the course, Tests and the Project must be passed (grade 2). One of the Test can be taken again.

Grade

$Jegy = (Test1 + Test2 + 2 \times Project) / 4$

Project topics

1. PlatypOUs



1.1. PlatypOUs path following

- **Basic:** Simulator setup, testing SLAM. Implementation of ROS node(s) to read the sensor data and move the robot.
- **Advanced:** Implementation of a ROS system for path follwoing in the simulator, using any sensor of the robot (e.g., driving next to the wall with given distance using LIDAR).
- **Epic:** Implementation and testing on the real robot/impress me!

1.2. PlatypOUs obstacle avoidance

- **Basic:** SSimulator setup, testing SLAM. Implementation of ROS node(s) to read the sensor data and move the robot.
- **Advanced:** Implementation of a ROS system to detect obstacle. Calculation and execution of a trajectory avoiding the obstacle in the simulator, using any sensor of the robot.
- **Epic:** Implementation and testing on the real robot/impress me!

1.3. PlatypOUs object follwoing

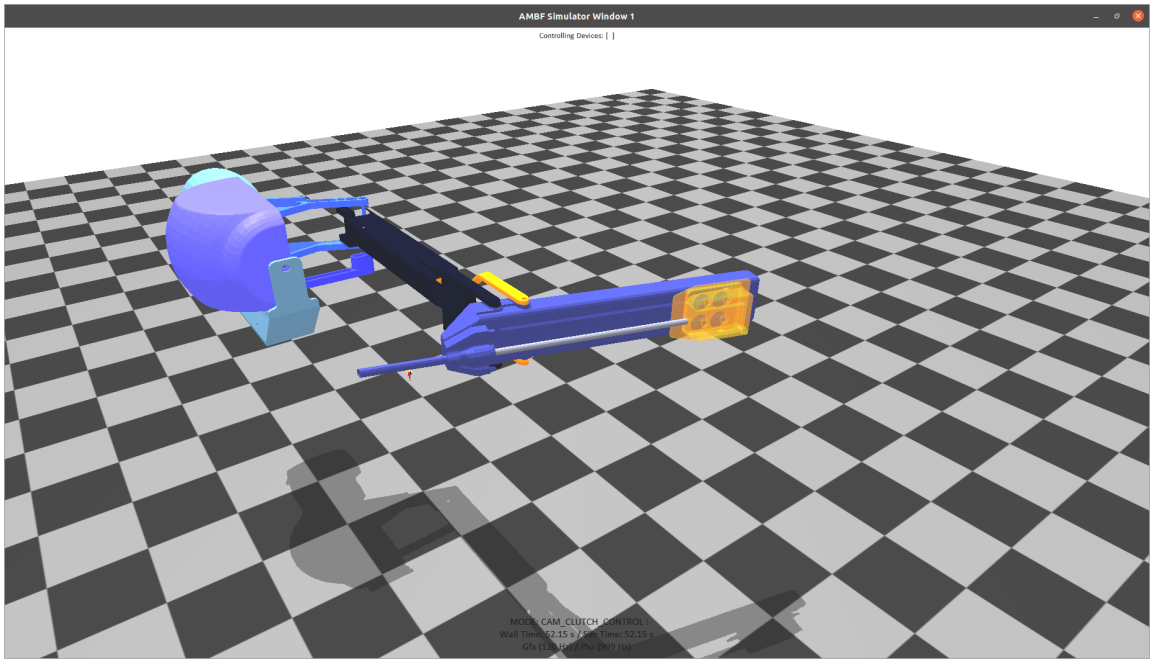
- **Basic:** Simulator setup, testing SLAM. Implementation of ROS node(s) to read the sensor data and move the robot.
- **Advanced:** Implementation of a ROS system to detect an object and follow it in the simulator, using any sensor of the robot(e.g., visual servoing).
- **Epic:** Implementation and testing on the real robot/impress me!

1.4. PlatypOUs action library

- **Basic:** Simulator setup, testing SLAM. Implementation of ROS node(s) to read the sensor data and move the robot.
- **Advanced:** Implementation of a ROS action library containing simple actions and their execution (e.g., push object, move to object, turn around).
- **Epic:** Implementation and testing on the real robot/impress me!

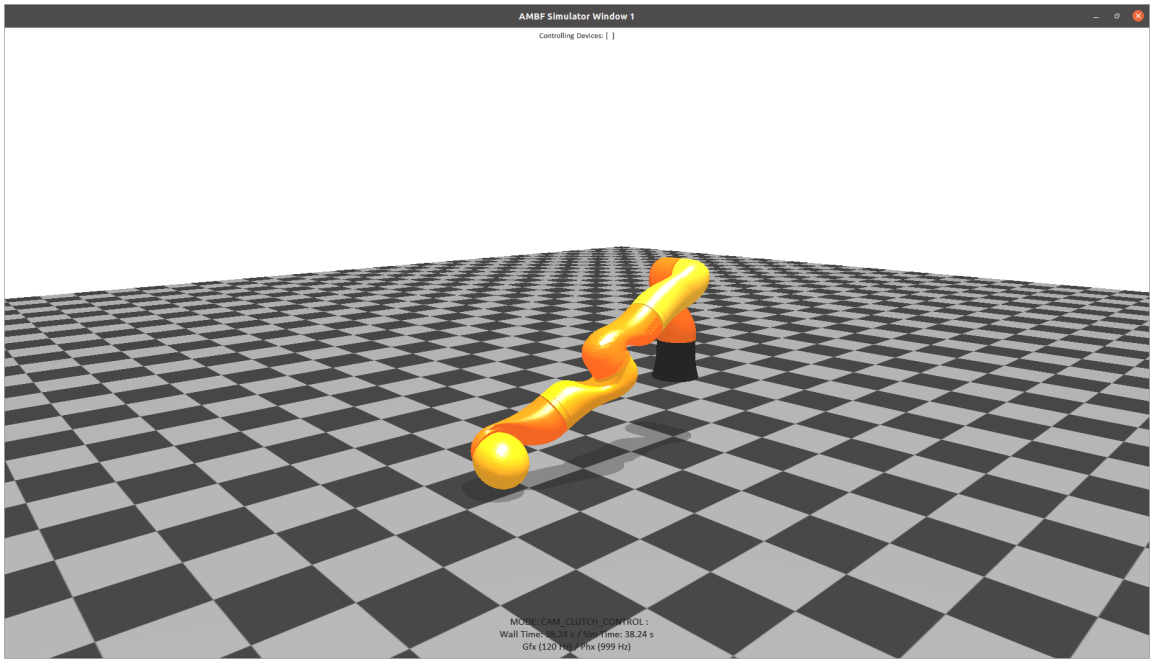
2. AMBF

2.1. ROS of a da Vinci robot in the AMBF simulator



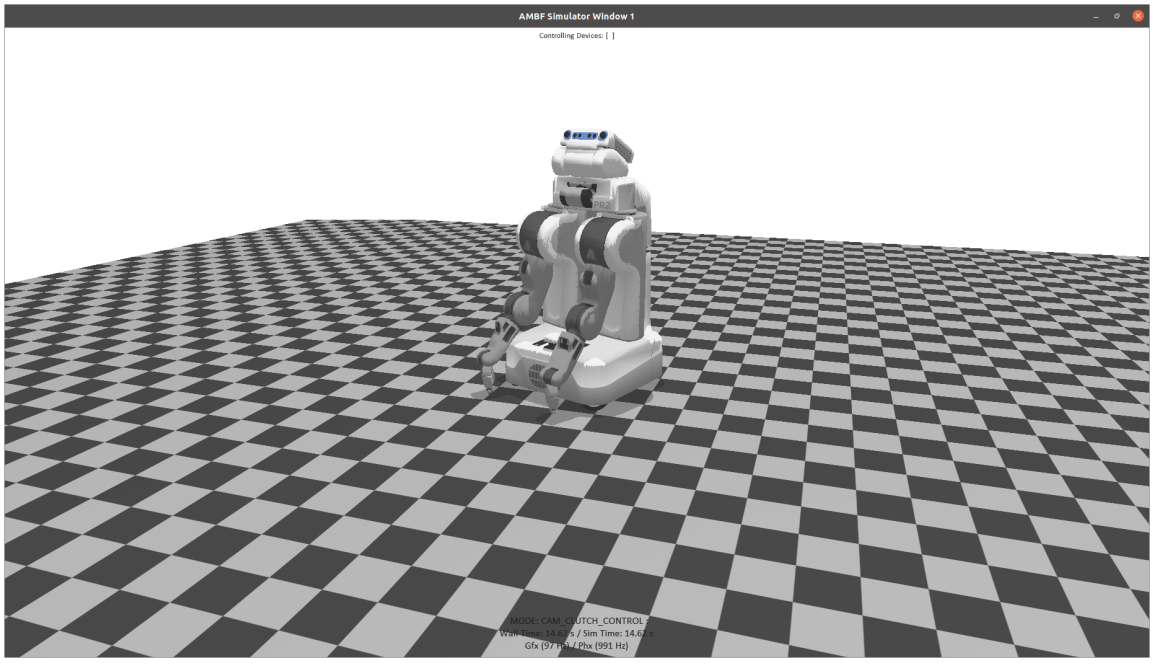
- **Basic:** Simulator setup, controlling the robot in joint space and task space (IK is implemented in AMBF) from ROS using the topic from CRTK.
- **Advanced:** Object detection in the *Peg transfer puzzle*.
- **Epic:** Autonomous manipulation in *Peg transfer*/implress me!

2.2. ROS integration of a KUKA robot in the AMBF simulator



- **Basic:** Simulator setup, controlling the robot in joint space from ROS.
- **Advanced:** Controlling the robot in task space, IK.
- **Epic:** Trajectory planning.

2.3. ROS integration of a PR2 humanoid robot in the AMBF simulator



- **Basic:** Simulator setup, controlling the robot in joint space from ROS.
- **Advanced:** Controlling the robot in task space, IK.

- **Epic:** Trajectory planning/Navigation/Manipulation.

X. Custom topic

Based on discussion.

Links

- Gazebo ROS packages
- PlatypOUs
- AMBF
- My fork of AMBF
- CRTK topics
- Navigation stack
- Paper on LiDAR SLAM
- Paper on vSLAM
- Paper on Visual Servoing Mobile Robot