Practical Exercise 2: Spatial Transformations

Objective:

In this practical exercise you should learn and apply spatial transformations and forward kinematics. For this purpose you implement the upcoming tasks in Python (here we provide a source code framework) or C++ (here, we **do not provide** a source code framework) and compare your implemented results with the output from CoppeliaSim. Ideally, they should be the same.

Task 1:

Start the program CoppeliaSim and load the scene ***euler_angle.ttt***. You will see only one reference frame. Find out which kind of euler angles are used in CoppeliaSim (order of elementary rotations). Calculate the total rotation matrix based on your findings on paper (two 3-by-3 matrix multiplications).

Additionally find out, how quaternions are implemented in CoppeliaSim. You can use the file ***P2_Task1.py*** for that purpose.

Implement the following functions (see Python file):

```
###########################################################################
# this needs to be implemented
def convert_quaternion_to_matrix(quat):
    # First row of the rotation matrix
    …..
###########################################################################
# this needs to be implemented
def euler_from_matrix(R):
    ……
###########################################################################
# this needs to be implemented
def matrix_from_euler(angles):
    ….
```

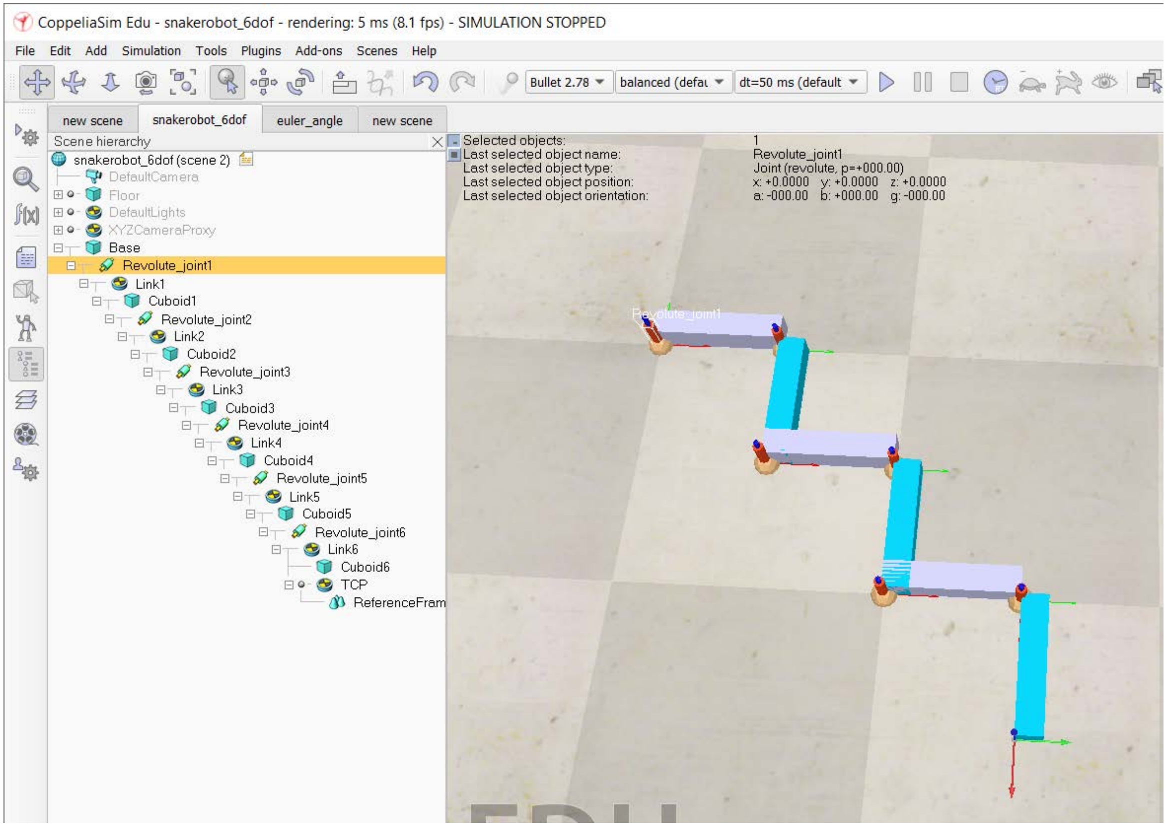
Compare your results with output from CoppeliaSim. They must be the same!

Task 2:

Build a “snake” type robot in a 2D plane consisting of 6 rotational joints, rotating around the z-axis of the world coordinate system (WCS).

To connect the joints by links, you can use a primitive shape like a cuboid in size 0.25, 0.05, 0,05 in x,y and z. Ensure, that these cuboids are static (untick dynamic property). Also ensure that the joints are in passive mode. The snake robot in its zero position (all joints are zero) should look as in figure 1.

Name all the objects you use the same way, especially the **joints** (Revolute_joint1, …) and the TCP frame (TCP), otherwise the python framework ***P2_Task2.py*** does not work.


CoppeliaSim Edu - snakerobot_6dof - rendering: 5 ms (8.1 fps) - SIMULATION STOPPED
File Edit Add Simulation Tools Plugins Add-ons Scenes Help
Bullet 2.78
balanced (defau
dt=50 ms (default
new scene
snakerobot_6dof
euler_angle
new scene
Revolute_joint1
EDU

```
Scene hierarchy
snakerobot_6dof (scene 2)
  DefaultCamera
  Floor
  DefaultLights
  XYZCameraProxy
  Base
    Revolute_joint1
      Link1
        Cuboid1
          Revolute_joint2
            Link2
              Cuboid2
                Revolute_joint3
                  Link3
                    Cuboid3
                      Revolute_joint4
                        Link4
                          Cuboid4
                            Revolute_joint5
                              Link5
                                Cuboid5
                                  Revolute_joint6
                                    Link6
                                      Cuboid6
                                      TCP
                                        ReferenceFram
```

```
Selected objects:                    1
Last selected object name:           Revolute_joint1
Last selected object type:           Joint (revolute, p=+000.00)
Last selected object position:       x: +0.0000   y: +0.0000   z: +0.0000
Last selected object orientation:    a: -000.00   b: +000.00   g: -000.00
```

Figure 1: snake robot with 6 rotational joints in its zero position

Task 3:

Extend the python script ***P2_Task2.py*** to build the forward kinematic chain for the snake robot starting from the WCS to the tip (TCP coordinate system).

Do not hardcode the forward kinematic chain with respect to only this snake robot. Implement a generic solution implementation, which would also work for other types of serial kinematic chains, see also the preparation of task 2.

Compare your results with output from CoppeliaSim. They must be the same!

Preparation:

Task1:

Calculate the total rotation matrix based on your findings on paper.

Task2:

Fill out the following table for the transformation parameters for the 6-dof snake robot. In the column DOF you should which kind of degree of freedom this joint is (Rx or Ry or Rz for rota-tional joints and Tx or Ty or Tz for translational joints.

Joint	DOF	Tx	Ty	Tz	Rx	Rx	Rx	Remark
	-							from world to robot base system
1	Rz							from 1. to 2. joint
2								from 2. to 3. joint
3								from 3. to 4. joint
4								from 4. to 5. joint
5								from 5. to 6. joint
6								from 6. Joint to TCP system

Hint: To transform from one joint to the next joint, first a **translation** is performed followed by a **rotation**.

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 file 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:

- You should name all files with you made with your names.
  - ZIP: name1_name2_exercise2.zip
    - PDF: name1_name2_exercise2.PDF
    - ….