5.2 Robot arm simulations

Since we did not always have access to the robot arm, we made a simulation to figure out how the UR10 would respond to our commands. We did this in anaconda spyder using the OpenGL and Pyqtgraph libraries. First, we made a representation of the UR10 with the same joint lengths and the same rotational direction of the joints. After the robot arm was implemented in the simulation, the forward kinematics were tested. The green sphere in the image shows the coordinates that the endeffector should be at after an input of all six joint angles. After a succesfull implementation of the forward kinematics, it was time to start the inverse kinematics, starting at the arm of the robot. We used a coordinate as an input to get the angles needed for the three joints of the arm. The green orb in the image is the input coordinate to which the robot arm needs to navigate to. In chapter 4.2.b is more information of the mathematics behind this control program. Next we add the wrist to the equation and we have the full simulation of the UR10. In the end we have the coordinate of the end effector as an input plus the pose that the wrist needs to take and we get the angles for all joints as an output. The UR10 has a python library called URX, which allows the user to control the movement of the UR10 using the libraries commands. For our project we will only be sending angles to the joints, to avoid getting acustomed to the overload of functionality that this universal robot has. We want to be able to implent our findings into a robot arm designed/chosen by ourselves without having to depend on the already implemented functions of this robot.
Program link (H4.2.b)