Project Intelligent Robotics
Assignment #1

In this assignment you will setup the software framework ROS we use throughout this course. You will get to know the fundamentals of the framework using the tutorial-simulator "`turtlesim`". The goal is to draw a "Haus vom Nikolaus" (an old german drawing puzzle for kids) in the simulator.

NOTE: We will use the ROS distribution "noetic" in this course. So always replace "<ROSVERSION>" with "noetic"!!!

Task 1.1 Set up ROS environment: Software for ROS is usually developed in **workspaces**. You will develop ROS **packages** in such a workspace in the next tasks, but first of all you have to setup the workspace itself.

1.1.1: Boot your Ubuntu system.

1.1.2: Create a folder you can use as your ROS workspace. In a terminal, type:

```
mkdir -p ~/ros/src
```

1.1.3: Next, turn this folder into a workspace. To do so, first you have to load the system ROS workspace that should be already installed:

```
source /opt/ros/<ROSVERSION>/setup.bash
```

Now you can initialize your own workspace on top of that one:

```
cd ~/ros
catkin build
```

1.1.4: Now, you could already load your new workspace by `source`'ing the workspace `setup.bash`. However, to ease development, you can also make your account load the workspace for you whenever you open a new terminal. To achieve this, add the following line to the end of the file `~/.bashrc` .
Do not just execute it in the terminal

```
source ~/ros/devel/setup.bash
```

1.1.5: Finally restart your shell to load the workspace in your current terminal:

```
exec bash
```

Task 1.2 Setup git: Working in groups or projects requires the use of a version control system. We are going to use git in this lecture. If you are already familiar with git you can quickly go through this task. If you are not, please look further into it as a homework. We will not give a full tutorial here, but do not hesitate to ask any question if you are unsure about git usage in specific situations!

1.2.1: Login to the Mafiasi Gitea (`https://git.mafiasi.de`) with your Mafiasi account (something like `11musterm`). If you don't have an account yet, create one at `https://mafiasi.de`. Make sure you are part of the `ir-project2023` team:
`https://git.mafiasi.de/org/TAMS/teams/ir-project2023`
If you are not, tell us to add you.

1.2.2: Go to `~/ros/src` and clone this existing git repository:

```
git clone https://git.mafiasi.de/TAMS/project23-tutorial
```

1.2.3: Create a new file ⟨YourName⟩.txt in the repository folder, commit and push it to the server.

```
git add <YourName>.txt
git commit
<type commit message in editor, save and exit>
git push
```

1.2.4: Pull the commit of the other participants, add your name as a line to one of the other files and commit and push the change. Resolve arising git conflicts peacefully talking to the others.

```
git pull
```

Task 1.3 Start turtlesim: In this section you will learn how to start ROS and ROS nodes. You will teleoperate a turtle in turtlesim and get to know basic ROS commands.

1.3.1: Open 4 terminals.
Start a roscore in the 1st one:

```
roscore
```

Start the turtlesim node in the 2nd:

```
rosrun turtlesim turtlesim_node
```

This brings up the turtlesim environment with one turtle in it. To teleoperate this turtle you can start another node to send commands for the turtle in the 3rd terminal:

```
rosrun turtlesim turtle_teleop_key
```

turtlesim should be started and you should be able to move the turtle around by using the arrow keys.

Now try some of these commands in the 4th terminal to learn about the environment you just started:

```
rosnode list
rosnode info /turtlesim

rostopic list
rostopic info /turtle1/pose
rostopic echo /turtle1/pose (Ctrl+c to exit)
```

```
rqt_graph (show "dead sinks" and "leaf topics")
rosservice call /kill "name: 'turtle1'"
<update in rqt_graph>
rosservice call /spawn <press tab,tab to autocomplete a spawn request>
<update in rqt_graph>
```

To teleoperate a turtle you spawn with the last command, you have to have the `turtle_teleop_key` node publish its commands to the `cmd_vel` topic of this turtle instead of `turtle1`:

```
rosrun turtlesim turtle_teleop_key turtle1/cmd_vel:=<name of your turtle>/cmd_vel
```

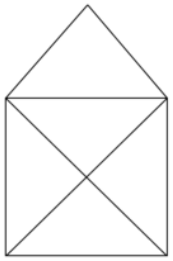
, e.g.,

```
rosrun turtlesim turtle_teleop_key turtle1/cmd_vel:=turtle2/cmd_vel
```

Task 1.4 Explain ROS: You should be able to explain these basic concepts of ROS:

- Nodes
- Topics
- Messages
- Services
- Actions
- Frames
- roscore

Task 1.5 Create your first ROS node: Write a node that moves the turtle around, such that it draws a ”Haus vom Nikolaus”. It looks like this and has to be drawn in one move.



1.5.1: All ROS nodes are part of **packages** to group similar functionality and keep track of dependencies. Create a ROS package with your/your group's name in the `project23-tutorial` repository. For this task, look up this tutorial:

`http://wiki.ros.org/ROS/Tutorials/CreatingPackage`

Note that ‘catkin‘ is the main build system for ROS and it is used to build the whole workspace. Your program will depend on the ‘turtlesim‘ and the ‘geometry_msgs‘ packages.

1.5.2: Look up the ROS tutorial for a simple publisher. There is one for C/C++ and one for Python. **You have to make sure you look at the tutorial for the right version of ROS**.

C/C++: `http://wiki.ros.org/ROS/Tutorials/WritingPublisherSubscriber(c++)`
Python: `http://wiki.ros.org/ROS/Tutorials/WritingPublisherSubscriber(python)`

1.5.3: Write a node (either in C++ or Python) that publishes messages that make the turtle draw a ”Haus vom Nikolaus”.

Every package for ROS has to be build at least once in order to make it known to the workspace. This has to be done even if you implement your node in python. Afterwards you can start your node with this command (just as you started the nodes in the previous task):

```
rosrun <ROS_PACKAGE> <THE_PROGRAMM>
```

Task 1.6 Using the git: Commit your program and get the packages of the others.

1.6.1: Inside the `project23-tutorial` folder, check for changes in your repository. There should be some, because you created your new package there.

```
git status
```

1.6.2: Now add your package and commit. Push your changes so the others can get it and pull the packages of the others participants.

```
git pull
git add <your package folder>
git commit
git push
```

1.6.3: Try out the nodes of the other participants and compare them to your own.

1.6.4: Read up on the ROS Services available in Turtlesim. You already learned about two of them earlier.

Select another working package in the repository and change the code of this project.

Instead of drawing the image once, the turtle should now be teleported to the beginning and re-draw the image. Bonus points for randomly changing the color of the house in each loop iteration.