6) BUILD YOUR OWN WORLD FOR NAVIGATION IN GAZEBO

We have seen how to work with Nav2 and what it is, by using existing simulation worlds

↓

we want the robot to move into a custom world.

Building a simulation of your real world is very useful to test your software application

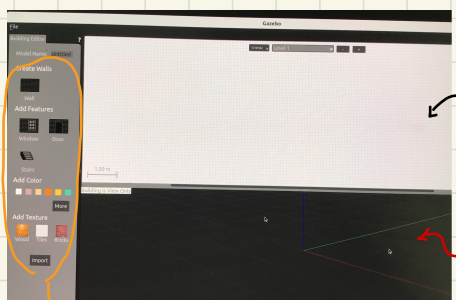
BUILD AND SAVE A WORLD IN THE GAZEBO BUILDING EDITOR

First, just run gazebo on its own: `gazebo`

↓

there, go to > Edit > Building Editor

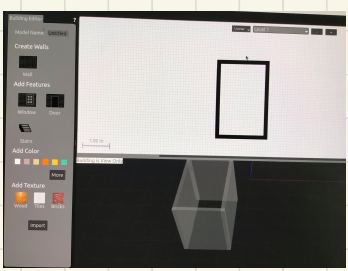
this space allow you to build a world with many elements



top-view (2D plan)

perspective view

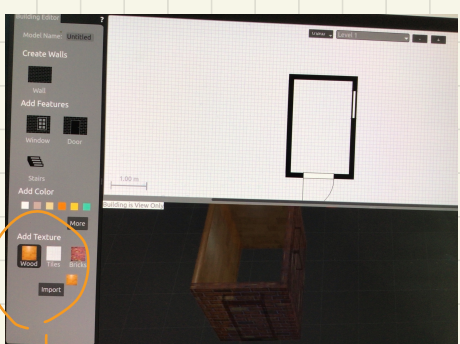
click here to add and customize elements



for example, adding 4 walls in a rectangular shape

then we can add doors, windows, textures and so on...

very easy world creation (even if now doors seems closed, it will be an opening wall later on)



add textures by clicking on the desired texture, then select element to apply it in the world 3D

+ adding stairs and create multiple levels is possible!

Then, To save this wored: File > save as > <name>

⇓

it will be saved in a new folder by default

it will generate two files: <name>.sdf , <name>.config

<name>.sdf: XML file, sdf is the wored description

<name>.config: just provide some model information

⇓

NOW, to import the wored created in gazebo, stunt it and then:

> Insert > path to custom wored > model_name

IF you don't find this path, click to "Add Path" optiom in Insert

↳ and you can drag and drop this as many time as you want

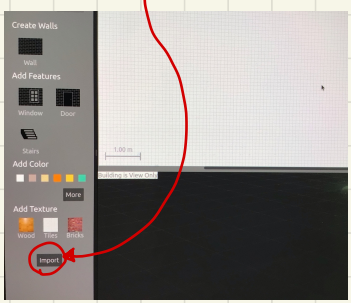
IMPORT A FLOOR PLAN

Building the wored from scratch is useful, but it is possible to start from a given floor plan.

this has all the rooms dimensiom

↓

we can import an image in the Gazebo wored editor, and start from that to build your wored



click on import and select the file, after selecting it, we need to set scale: to know how big are the image dimensiom

to set scale: use two points with known distance.
Just click on two points with a distance in meters that you know, and set the distance ↳ this will fix the entire scale

IF you already have a scale factor metric, you can use that to set the plane scale.
↓
finally click "OK" ↝ you will see that floor plan in 2D view

Now we can start to build on top of the image, using walls, doors and windows
(by holding SHIFT during wall construction I have more precision)

By building on top of the floor plan, with given scale, we will obtain a 3D world of the correct dimension

ADD OBJECTS TO THE WORLD

we build a custom world with rooms / doors / windows...
What about objects?
↳ start gazebo and insert your custom model plan
⇓
Then, on the Insert tab, we have different possible path from which insert models.
FROM https://models.gazebo... we can add common objects to the world
(sometimes it may take a while or crash when adding objects)

Add the object you want in the desired positions, Then, to save this world you can go to

>File> save World As> ... save it with <world_names.world

while in the building editor we save it with ".sdf", ".config" extensions, When saving as Gazebo world, we save with ". world" extension

this is also an xml world description (sdf)

⇓

I can start gazebo with this custom world by launching it as

gazebo <world_name>.world

MAKE TURTLEBOT3 NAVIGATE IN THAT WORLD

STEP 1, CREATE TURTLEBOT3_GAZEBO OVERLAY

Once we create our custom world, we can use this simulated world when testing our Robot

GOAL: Adapt launch file so that we can spawn turtlebot 3 in this custom world and test Navigation / mapping on this world.

to run turtlebot3 in a simulated environment, we launched

turtlebot3_gazebo turtlebot3_world.launch.py

the launch configuration is in this package

go to /opt/ros/humble/share/turtlebot3_gazebo/worlds

↑ here I have all installed packages

⇓

we need to adapt this package!

BUT we are not going to modify it where it is installed, Instead we will clone it in our own workspace

for example

- ros2_ws then on it follow usual steps to create a workspace

We will clone in ros2_ws/src/ the turtlebot3_gazebo package and override it!

so, we will have the underlay in /opt/ros/humble/share
while the overlay in /ros2_ws/src
↑
we create an overlay of turtlebot3_gazebo package, to modify it and use it, as well as call in the underlay

find turtlebot3_gazebo in github → we will have to clone turtlebot3_simulations containing turtlebot3_gazebo
↓
use git clone as usual in /src

git clone <URL>

Then, we need to checkout the correct branch, with your ROS2 distribution

git checkout humble-devel
your ros version

Now you can colcon build the workspace from /ros2_ws

(remember to set-up the workspace, so let's add the
source ~/ros2_ws/install/setup.bash)
↑ this has to be AFTER the source /opt/ros/humble/setup.bash to work correctly!

⇓
by launching turtlebot3_gazebo, now the launcher will be loaded NOT from global ros installation, but from our own workspace
↳ we can modify and customize it in our overlay!

STEP 2, ORGANIZE CUSTOM LAUNCH FILE FOR OUR Gazebo WORLD

Let's modify the turtlebot3_gazebo package to use our custom world and start turtlebot 3 robot there.

Looking in the /launch folder, we have to create our own launch file there.
And a parameter loaded from /worlds define which world to load
we need 2 steps:

1) Add custom world
2) New launch file

1) Move custom world to .../worlds

move <world>.world inside turtlebot3_gazebo/worlds
(in our overlay)

↓

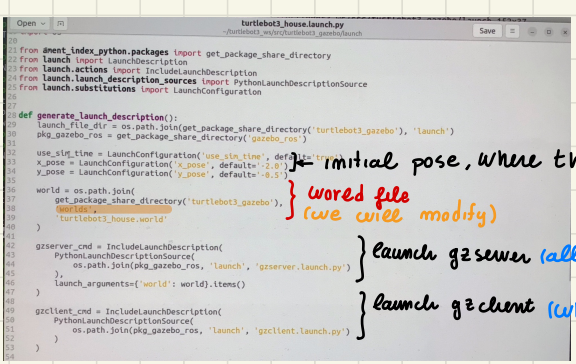
to make it appear as the other .world file, add the
<?xml version="1.0" ?> at the beginning of the file

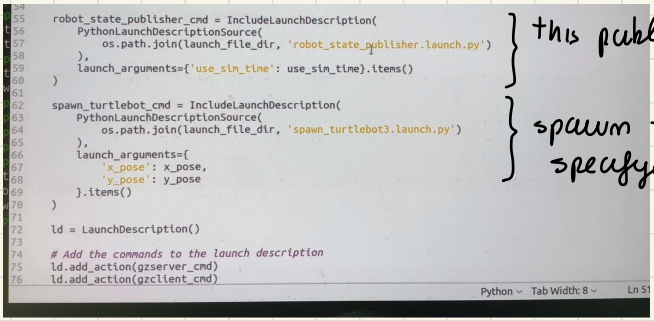
so, also
it will be recognized as xml by text editor

2) Define a custom launch file

We will copy one existing turtlebot3_gazebo launcher and properly modify its parameters

For example: turtlebot3_house.launch.py





```
robot_state_publisher_cmd = IncludeLaunchDescription(
    PythonLaunchDescriptionSource(
        os.path.join(launch_file_dir, 'robot_state_publisher.launch.py')
    ),
    launch_arguments={'use_sim_time': use_sim_time}.items()
)

spawn_turtlebot_cmd = IncludeLaunchDescription(
    PythonLaunchDescriptionSource(
        os.path.join(launch_file_dir, 'spawn_turtlebot3.launch.py')
    ),
    launch_arguments={
        'x_pose': x_pose,
        'y_pose': y_pose
    }.items()
)

ld = LaunchDescription()

# Add the commands to the launch description
ld.add_action(gzserver_cmd)
ld.add_action(gzclient_cmd)
```

Python Tab Width: 8 Ln 51

this publish TF for the robot

spawn turtlebot3 with specified x,y pose

⇓

we can copy this launch file in a new file

turtlebot3_my_world.launch.py

↳ then, edit it:

just edit the worlds by changing world name to my_world.world

NOW we can build the workspace!

(remember to source or open new terminal)

And you can test the custom world launching

turtlebot3_gazebo turtlebot3_my_world.launch.py

will start

ROBOT in the world, in the position x,y specified in our launch file in x_pose, y_pose

⇓

we can customize robot spawn position with desired initial position accordingly to our world (ex -1.0, -5.0 in the floor map)

⇓

Once turtlebot3 spawn in this custom world, we can move it with the teleop_keyboard node as before!

STEP 3, MAP AND NAVIGATE IN THE CUSTOM WORLD

Now robot is fully integrated in custom world.
We are ready to reproduce the mapping and navigation procedures explained on previous lectures

1) Launch simulation
2) run teleoperation
3) Launch SLAM with turtlebot3_cartographer
   ... at the end of map reconstruction, save the map with map_saver_cli

IF you have missing world parts it is possible to fix that by hand, in a post-process phase!
to make it more reliable for navigation
(we will see later how to do this)

NOTE: according to the 2D plane of Lidar, we may have some issue detecting some parts of the environment if at lower height, or due to visibility limit.
(you can clean the map later on)

CLOSE ALL TERMINALS, relaunch simulation
4) Launch Navigation tools with navigation2.launch.py passing the new map as argument
   - select 2D Pose estimate
   - send Nav2 Goal / waypoint

TIPS: HOW TO FIX AND IMPROVE MAPS WITH GIMP

once we reconstruct the map, it is possible to fix the missing map informations (missing pixels).

A solution can be to restart SLAM and remap, but it is time consuming...

- uncomplete wall
- free space still GREY unrecognized
- twisted map during SLAM

being a map
just an image (set of pixels)
→ to make it more accurate, we can just edit the image

This can be done with GIMP, free editing program for images
easy to use! you can use Photoshop or others if you want

sudo snap install gimp

then you can launch it with gimp
and modify the map.img by simply drag and drop the image in GIMP UI

- Few tips on GIMP usage:

  - CTRL + Mouse to zoom in/out

We just need black/white color to modify the map.pgm
by using Pencil tool, select the size properly

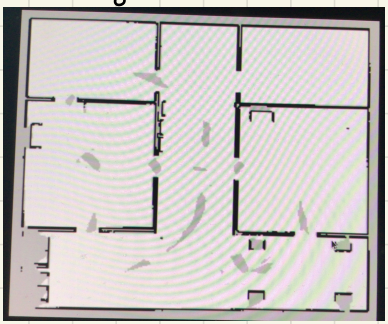
Black pencil: fix the missing wall pixels, objects perimeter etc.

complete wall
pixels closing lines, just for navigation purpose!
We don't need accurate lines

Also add elements in the ground NOT detected by LIDAR if you know about some undected obstacles

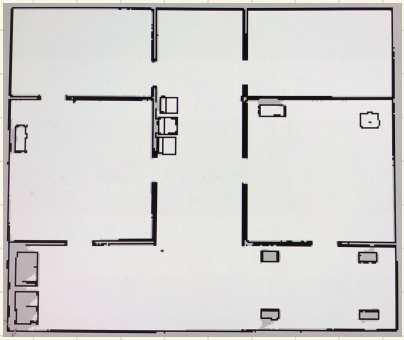
Then, using White pencil: clear the part of the map we know is free space, for wrong obstacles or unknown grey space.

before



⇒

AFTER GIMP post-process



>File > Export As ...

There is no need to rename / modify the <maps.yaml, be sure that the image parameter in configuration yaml file link to the new version of the map

↓

then, when launching navigation tools, the map loaded is the modified one

Another use of map editing with GIMP: when we have big environment, mapping ALL at once may be a problem...

→ we can make some maps of some areas and then merge more images into one

• ASSIGNMENT 3 · MAKE A ROBOT NAVIGATE INTO A GENERATED MAZE

- Create your own custom world
- Adapt turtlebot3 for custom world
- Make Robot navigate

} follow same steps as the ones of this lecture

We are going to generate map by a maze generator online

→ mazegenerator.net

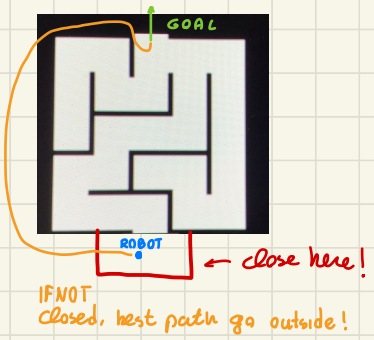
> Generate new and save as >PNG (NOT as PDF) Them, download

> 5×5 dimension is suggested, too big becomes hard to map

Assignment

1) Create world in Gazebo from maze as:
   - ".world" extension,
   - close one of the entrance at least → because if we want the Robot to traverse the maze, if we leave both dors open, it will go to the goal from the outside of the maze
   - Use 1.2m scale for door frame

you can add objects if you want!



2) Adapt turtlebot3_gazebo package to spawn turtlebot on this world
   - spawn it at one of the entrances

3) Generate map of maze with SLAM + Navigate on it

USEFULL COMMENTS ON THE SOLUTION

PART 1)
- Avoid excessive narrow spaces when recreating the map with Gazebo Building Editor. otherwise when navigating we will face issues during navigation.
- After creating the maze, generate a .world file from gazebo itself

PART 2) Just follow the same step to create a new launch file

PART 3)
- When you send a Nav2 Goal in unknown (grey) space, the Nav2 stack generate a straight trajectory to the goal, Unknown will be considered as free space.

- When navigating from start to end of the maze, the global path is easily found, but you can encounter issues when "U shape" manouvers has to be done.
  you can try to redefine a waypoint following over the planned path to complete navigation.