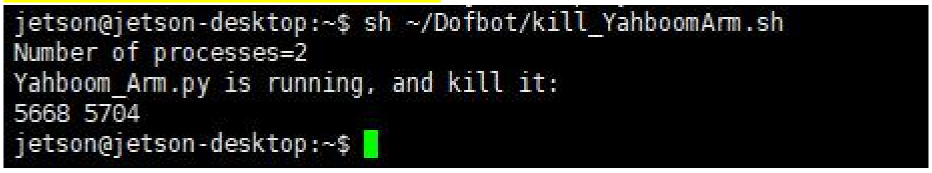
Enter dofbot's system through a virtual machine.
MoveIt case code path:/home/dofbot/dofbot_ws/src

Part-1. Open MoveIt simulation environment steps:
1. Close the APP remote control process temporarily.
Input following command:
sh ~/Dofbot/kill_YahboomArm.sh



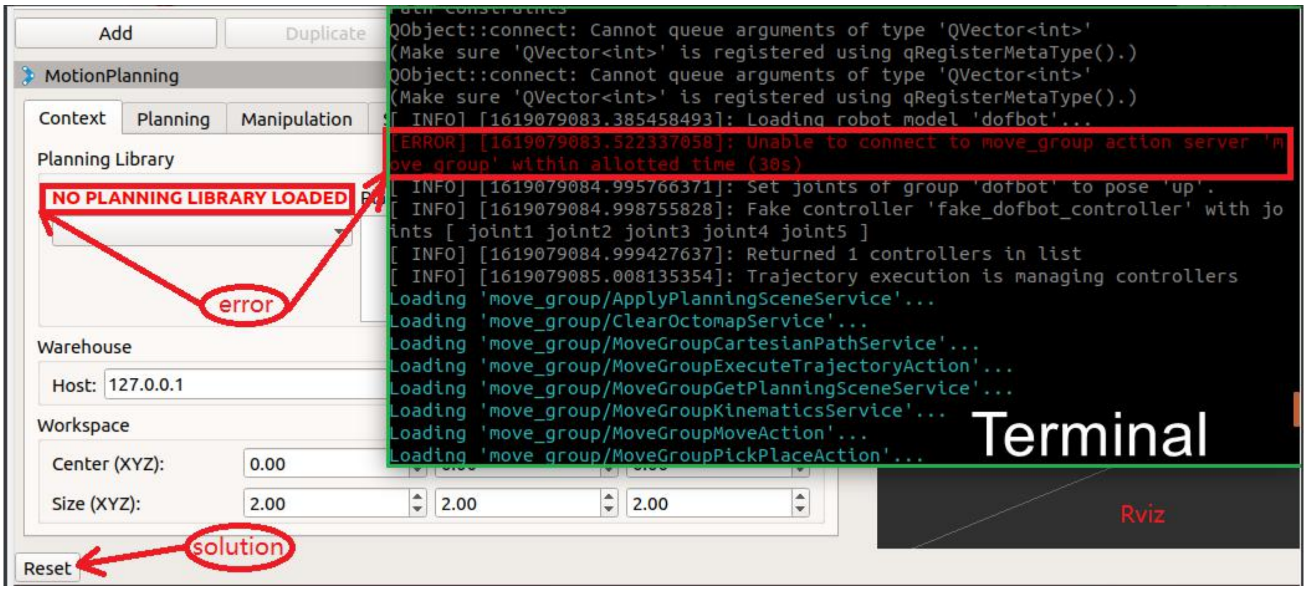
```
jetson@jetson-desktop:~$ sh ~/Dofbot/kill_YahboomArm.sh
Number of processes=2
Yahboom_Arm.py is running, and kill it:
5668 5704
jetson@jetson-desktop:~$
```

2. Open and update the MoveIt simulation environment and start a terminal.

```
cd dofbot_ws/
catkin_make
source devel/setup.bash
roslaunch dofbot_config demo.launch
```

(Keep running this simulation environment until the entire MoveIt case is over and then close it.)

The MoveIt simulation environment starts slowly, so wait patiently for a period of time.
In the process of loading the simulation environment, if the following prompt appears.
At the same time, the Rviz emulator also showed a prompt for no be loaded. As shown below.

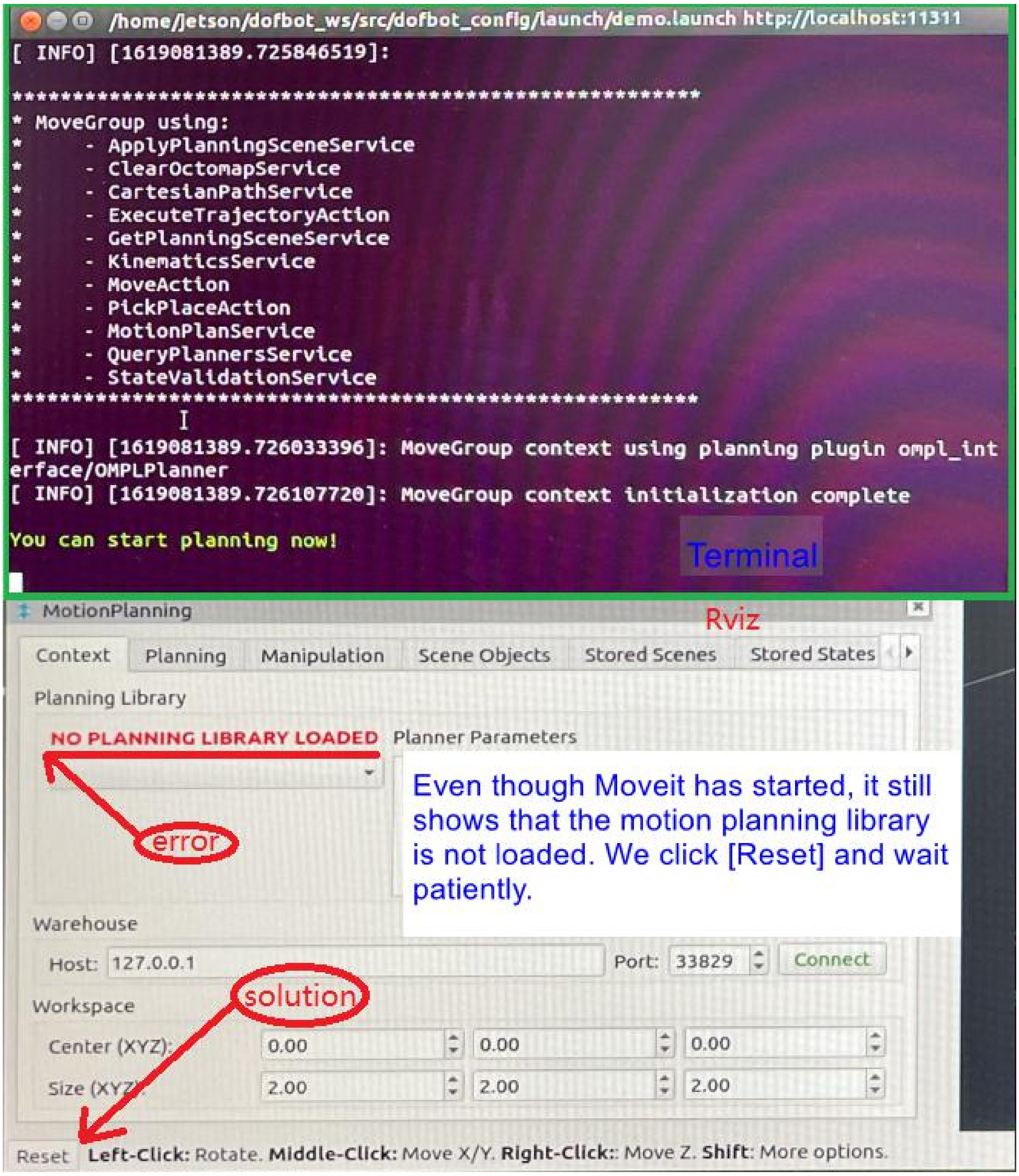


```
QObject::connect: Cannot queue arguments of type 'QVector<int>'
(Make sure 'QVector<int>' is registered using qRegisterMetaType().)
QObject::connect: Cannot queue arguments of type 'QVector<int>'
(Make sure 'QVector<int>' is registered using qRegisterMetaType().)
[ INFO] [1619079083.385458493]: Loading robot model 'dofbot'...
[ERROR] [1619079083.522337058]: Unable to connect to move_group action server 'm
ove_group' within allotted time (30s)
[ INFO] [1619079084.995766371]: Set joints of group 'dofbot' to pose 'up'.
[ INFO] [1619079084.998755828]: Fake controller 'fake_dofbot_controller' with jo
ints [ joint1 joint2 joint3 joint4 joint5 ]
[ INFO] [1619079084.999427637]: Returned 1 controllers in list
[ INFO] [1619079085.008135354]: Trajectory execution is managing controllers
Loading 'move_group/ApplyPlanningSceneService'...
Loading 'move_group/ClearOctomapService'...
Loading 'move_group/MoveGroupCartesianPathService'...
Loading 'move_group/MoveGroupExecuteTrajectoryAction'...
Loading 'move_group/MoveGroupGetPlanningSceneService'...
Loading 'move_group/MoveGroupKinematicsService'...
Loading 'move_group/MoveGroupMoveAction'...
Loading 'move_group/MoveGroupPickPlaceAction'...
```

We need to click [Reset] in the lower left corner to re-load.
Note: If you click [Reset] before loading, the system will reload, which will cause the system to fail to start completely.

As shown below.

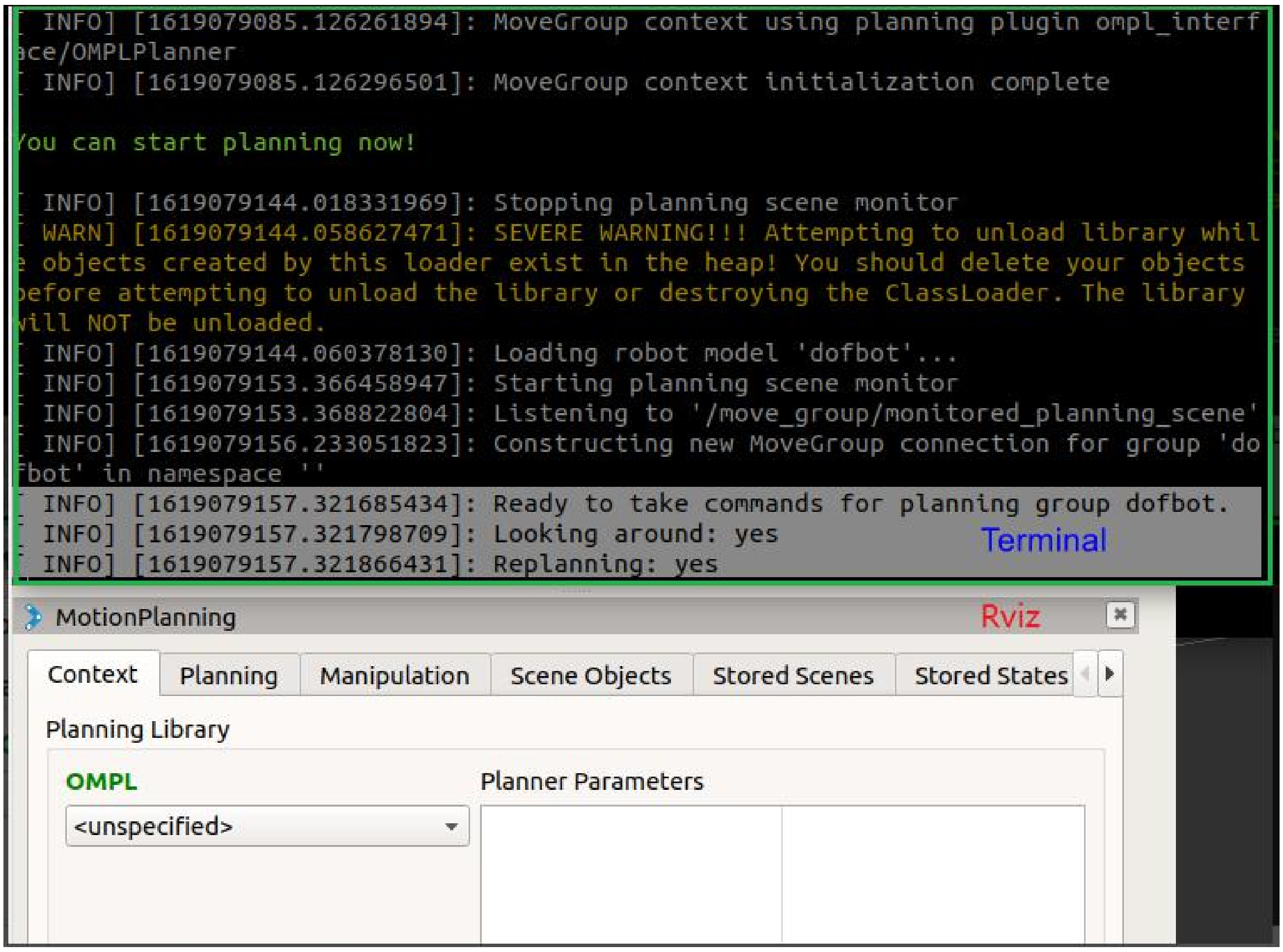
Even though Moveit has started, it still shows that the motion planning library is not loaded. We click [Reset] and wait patiently. It is related to the performance of the device, and sometimes it takes a few more attempts or restarts.



```
/home/jetson/dofbot_ws/src/dofbot_config/launch/demo.launch http://localhost:11311
[ INFO] [1619081389.725846519]:

**********************************************************
* MoveGroup using:
*     - ApplyPlanningSceneService
*     - ClearOctomapService
*     - CartesianPathService
*     - ExecuteTrajectoryAction
*     - GetPlanningSceneService
*     - KinematicsService
*     - MoveAction
*     - PickPlaceAction
*     - MotionPlanService
*     - QueryPlannersService
*     - StateValidationService
**********************************************************

[ INFO] [1619081389.726033396]: MoveGroup context using planning plugin ompl_int
erface/OMPLPlanner
[ INFO] [1619081389.726107720]: MoveGroup context initialization complete

You can start planning now!
```

Even though Moveit has started, it still shows that the motion planning library is not loaded. We click [Reset] and wait patiently.

As shown below, [Replanning: yes] appears on the terminal and a green [OMPL] appears under Planning Library, which means MoveIt simulation environment has been successfully started.



```
[ INFO] [1619079085.126261894]: MoveGroup context using planning plugin ompl_interf
ace/OMPLPlanner
[ INFO] [1619079085.126296501]: MoveGroup context initialization complete

You can start planning now!

[ INFO] [1619079144.018331969]: Stopping planning scene monitor
[ WARN] [1619079144.058627471]: SEVERE WARNING!!! Attempting to unload library whil
e objects created by this loader exist in the heap! You should delete your objects
before attempting to unload the library or destroying the ClassLoader. The library
will NOT be unloaded.
[ INFO] [1619079144.060378130]: Loading robot model 'dofbot'...
[ INFO] [1619079153.366458947]: Starting planning scene monitor
[ INFO] [1619079153.368822804]: Listening to '/move_group/monitored_planning_scene'
[ INFO] [1619079156.233051823]: Constructing new MoveGroup connection for group 'do
fbot' in namespace ''
[ INFO] [1619079157.321685434]: Ready to take commands for planning group dofbot.
[ INFO] [1619079157.321798709]: Looking around: yes
[ INFO] [1619079157.321866431]: Replanning: yes
```

Part-2. Execute the MoveIt case separately

1. Set movement and open another port.

Code path: /home/dofbot/dofbot_ws/src/dofbotmoveit/scripts/01set_move.py

Input following command on terminal.

```
cd dofbot_ws/
source devel/setup.bash
rosrun dofbot_moveit 01_set_move.py    # python file
# Or
rosrun dofbot_moveit 01_random_move # c++ file
```

Close case: [ctrl+c] or [ctrl+z]

About code:

#Import header file

```
import rospy
```

```
from moveit_commander.move_group import MoveGroupCommander
```

# Initialize node
```
rospy.init_node("set_move")
```

# Initialize the robotic arm motion group
```
dofbot = MoveGroupCommander("dofbot")
```
Note: The group name here must be the same as the group name in the previous MoveIt configuration, otherwise it will be invalid.

Set random target points
```
dofbot.set_random_target()
```

# Set posture
```
dofbot.set_named_target("up")
dofbot.set_named_target("down")
```
Note: Up and down are the postures set during MoveIt configuration.

Drive function.
```
dofbot.go()
```

2. Motion planning, reopen a port

Code path: dofbot_ws/src/dofbot_moveit/scripts/02_motion_plan.py

Input following command on terminal.

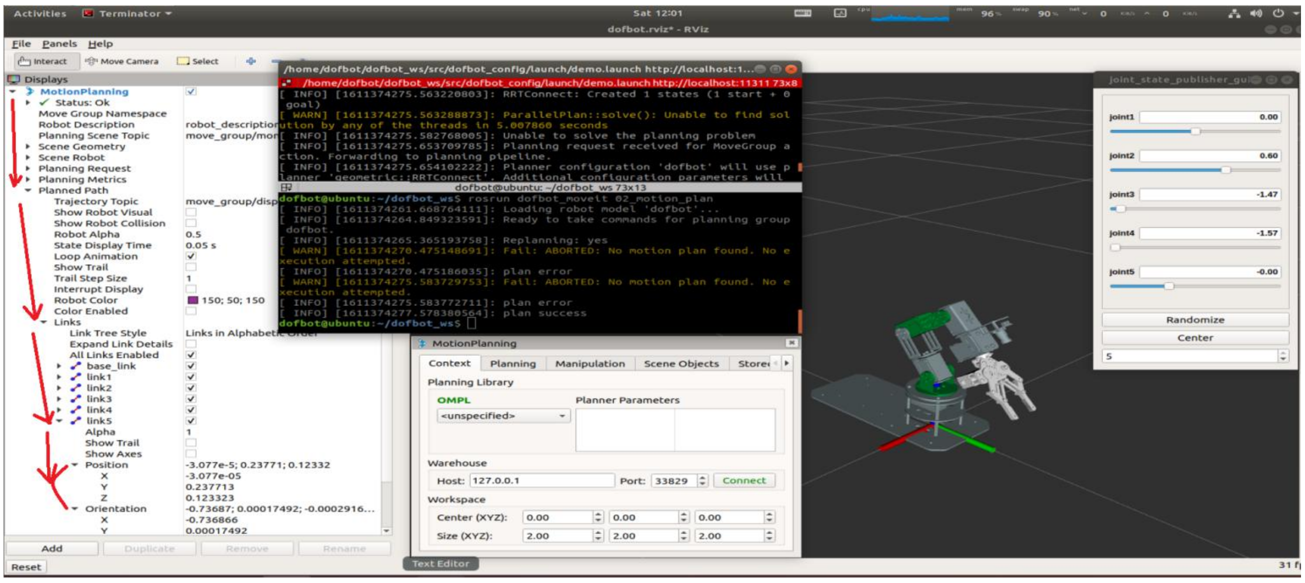
```
cd dofbot_ws/
source devel/setup.bash
rosrun dofbot_moveit 02_motion_plan.py    # python file
# Or
rosrun dofbot_moveit 02_motion_plan    # c++ file
```

Close case: [ctrl+c] or [ctrl+z]

As shown below, follow the red arrow on the left to view the end pose information (position and posture) in real time.

Position: Position; Orientation: Four elements represent posture.

In motion planning, not only the position reached by the robotic arm must be considered, but also the posture when reaching that position. Multiple planning can improve the success rate, and you can also consider replacing the kinematics plug-in.


```
/home/dofbot/dofbot_ws/src/dofbot_config/launch/demo.launch http://localhost:11311 73x8
[ INFO] [1611374275.563220803]: RRTConnect: Created 1 states (1 start + 0
 goal)
[ WARN] [1611374275.563288873]: ParallelPlan::solve(): Unable to find sol
ution by any of the threads in 5.007860 seconds
[ INFO] [1611374275.582768005]: Unable to solve the planning problem
[ INFO] [1611374275.653709785]: Planning request received for MoveGroup a
ction. Forwarding to planning pipeline.
[ INFO] [1611374275.654102222]: Planner configuration 'dofbot' will use p
lanner 'geometric::RRTConnect'. Additional configuration parameters will
```

```
dofbot@ubuntu:~/dofbot_ws$ rosrun dofbot_moveit 02_motion_plan
[ INFO] [1611374261.668764111]: Loading robot model 'dofbot'...
[ INFO] [1611374264.849323591]: Ready to take commands for planning group
 dofbot.
[ INFO] [1611374265.365193758]: Replanning: yes
[ WARN] [1611374270.475148691]: Fail: ABORTED: No motion plan found. No e
xecution attempted.
[ INFO] [1611374270.475186035]: plan error
[ WARN] [1611374275.583729753]: Fail: ABORTED: No motion plan found. No e
xecution attempted.
[ INFO] [1611374275.583772711]: plan error
[ INFO] [1611374277.578380564]: plan success
dofbot@ubuntu:~/dofbot_ws$
```

MotionPlanning	✓
Status: Ok	
Robot Description	robot_description
Planning Scene Topic	move_group/mon
Trajectory Topic	move_group/disp
Show Robot Visual	
Show Robot Collision	
Robot Alpha	0.5
State Display Time	0.05 s
Loop Animation	✓
Show Trail	
Trail Step Size	1
Interrupt Display	
Robot Color	150; 50; 150
Color Enabled	
Link Tree Style	Links in Alphabetic Order
Expand Link Details	
All Links Enabled	✓
base_link	✓
link1	✓
link2	✓
link3	✓
link4	✓
link5	✓
Alpha	1
Show Trail	
Show Axes	
Position	-3.077e-5; 0.23771; 0.12332
X	-3.077e-05
Y	0.237713
Z	0.123323
Orientation	-0.73687; 0.00017492; -0.0002916...
X	-0.736866
Y	0.00017492

About code:

Create a pose instance

```
pos = Pose()
```

Set pose information

```
# Setting position (unit: m)
pos.position.x = 0.0
pos.position.y = 0.05957241
pos.position.z = 0.1680498
# Set attitude RPY (unit: angle)
roll = -140.0
pitch = 0.0
yaw = 0.0
```

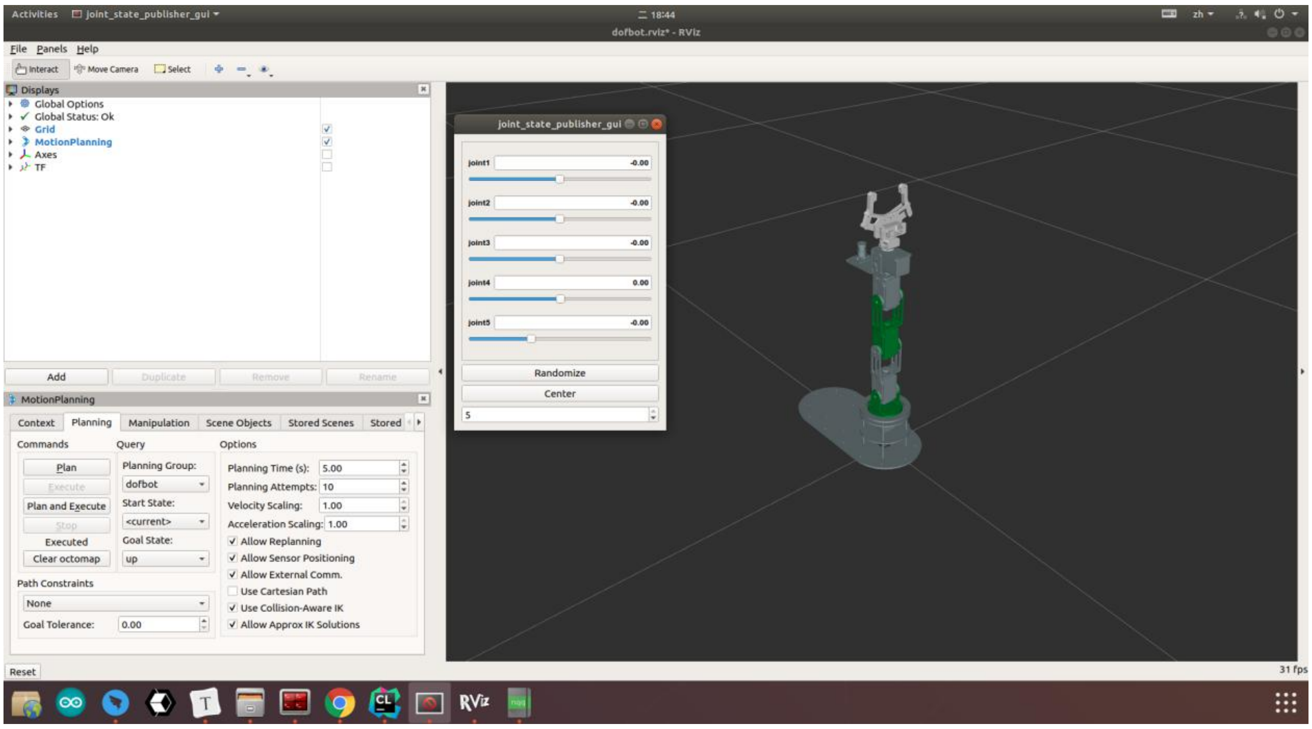
Add target piont

```
dofbot.set_pose_target(pos)
```

Execute plan

```
plan = dofbot.plan()
```

Rviz drive, as shown below.



Planning Group: dofbot
Start State: <current>
Goal State: up
Path Constraints: None
Goal Tolerance: 0.00
Planning Time (s): 5.00
Planning Attempts: 10
Velocity Scaling: 1.00
Acceleration Scaling: 1.00
✓ Allow Replanning
✓ Allow Sensor Positioning
✓ Allow External Comm.
☐ Use Cartesian Path
✓ Use Collision-Aware IK
✓ Allow Approx IK Solutions

joint1 -0.00
joint2 -0.00
joint3 -0.00
joint4 0.00
joint5 -0.00

【Start State】: start posture
【Goal State】: target posture
【Plan】: Motion planning
【Execute】: Plan first before execution
【Plan and Execute】: Plan and execute
Different target postures can be changed, click Plan to display the process of motion planning, click [Plan and Execute] to display and move to the target posture.

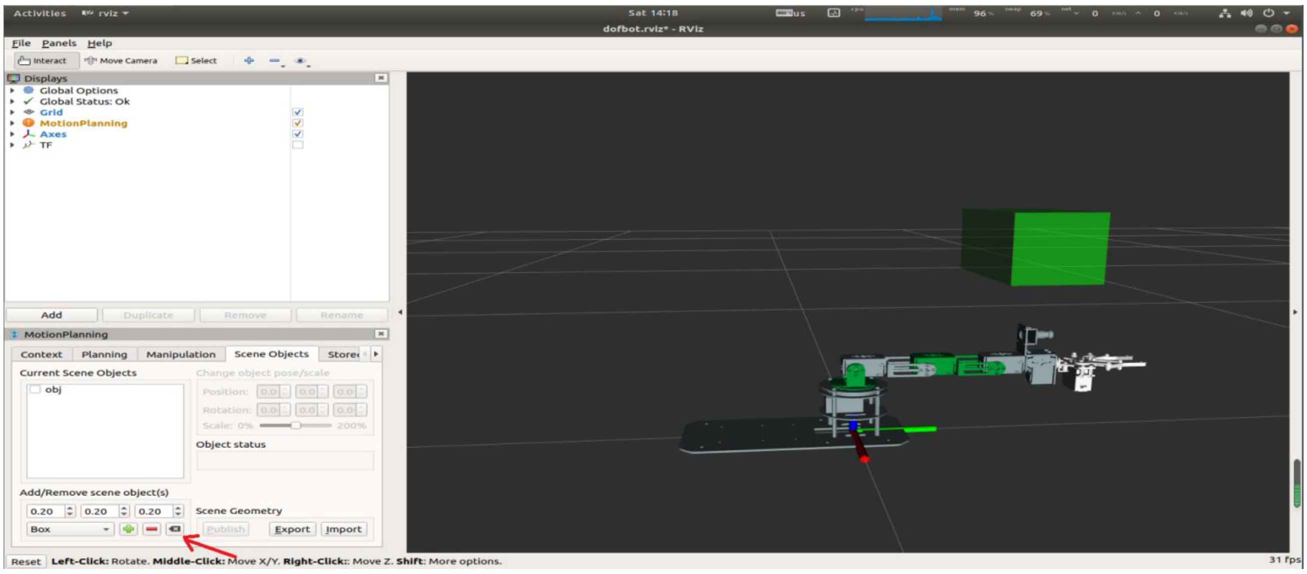
3. Impact checking

3.1 Method-Run the code we provided.

Code path: dofbot_ws/src/dofbot_moveit/scripts/03_attached_object.py
Input following command to start this function.

```
cd dofbot_ws/
source devel/setup.bash
rosrun dofbot_moveit 03_attached_object.py  # python file
# Or
rosrun dofbot_moveit 03_attached_object # c++ file
```

Close case: [ctrl+c] or [ctrl+z]
Click [x] to delete all obstacles.

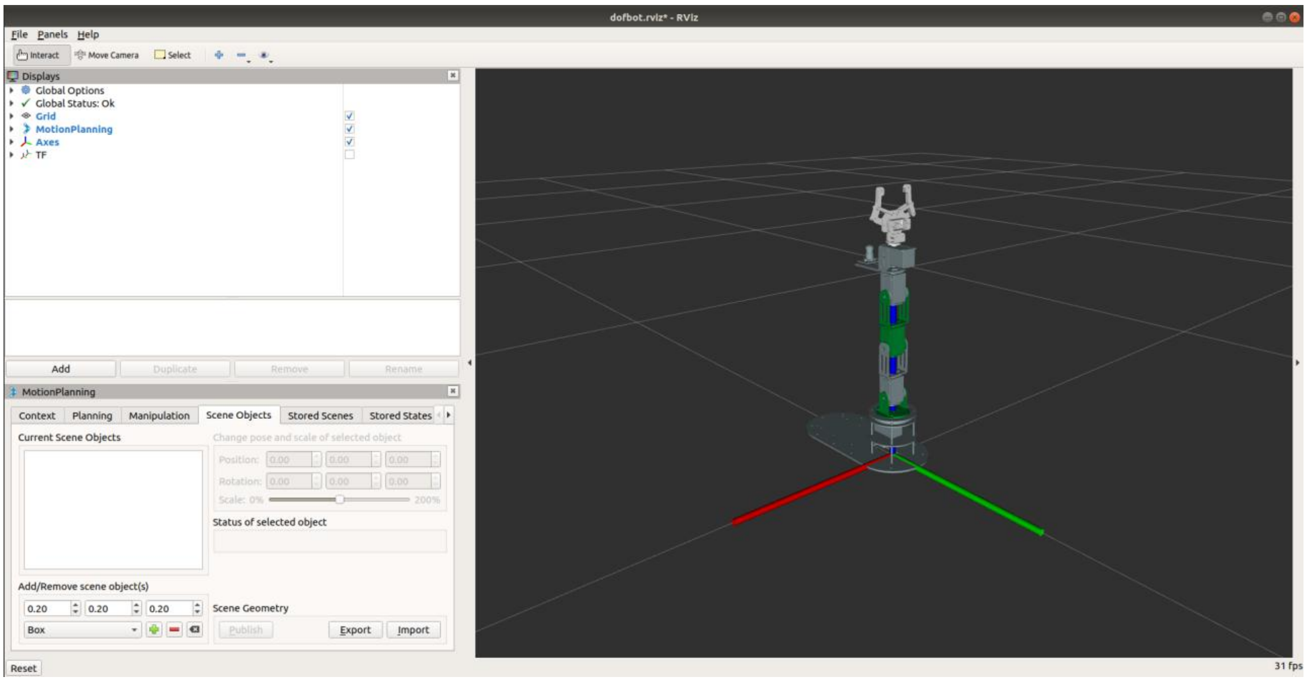


3.2 Run this function on rviz directly

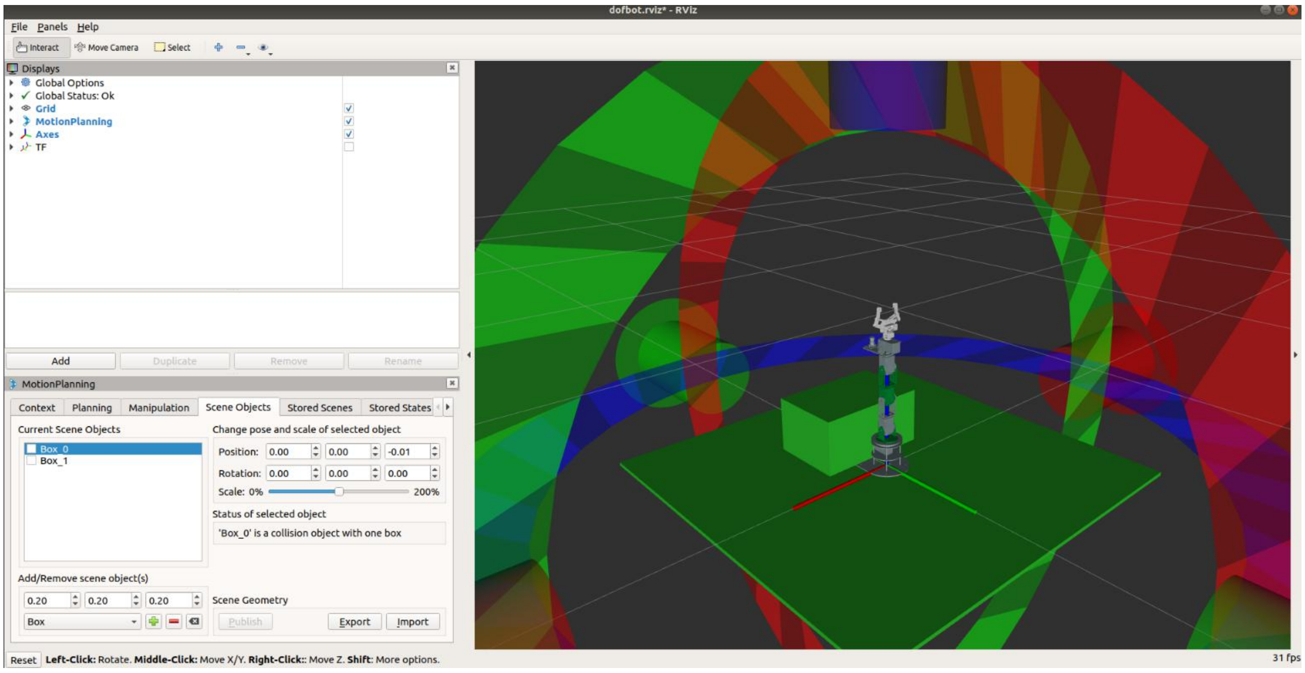
rviz directly created, as shown below.

Click “SceneObjects” in MotionPlanning to enter the scene settings, we select the Box property, adjust the scene object(s) parameters.

Click the “+” to add obstacles; click the “-” to delete obstacles.



Adjust the Position and Rotation properties, change the pose of the box, add Box_0 and Box_1 two boxes, click [Publish] after adding it to be effective; click the [Export] button to export the scene, save it to dofbot_ws/src/dofbot_config/scene /floor.scene; next time you use it, just click [Import] to import.


dofbot.rviz* - RViz
File Panels Help
Interact Move Camera Select
Displays
Global Options
Global Status: Ok
Grid
MotionPlanning
Axes
TF
Add Duplicate Remove Rename
MotionPlanning
Context Planning Manipulation Scene Objects Stored Scenes Stored States
Current Scene Objects
Change pose and scale of selected object
Status of selected object
Add/Remove scene object(s)
Scene Geometry
Publish Export Import
Reset
31 fps

Box_0
Box_1

Position: 0.00 0.00 -0.01
Rotation: 0.00 0.00 0.00
Scale: 0% 200%

'Box_0' is a collision object with one box

0.20 0.20 0.20
Box

Left-Click: Rotate. Middle-Click: Move X/Y. Right-Click:: Move Z. Shift: More options.

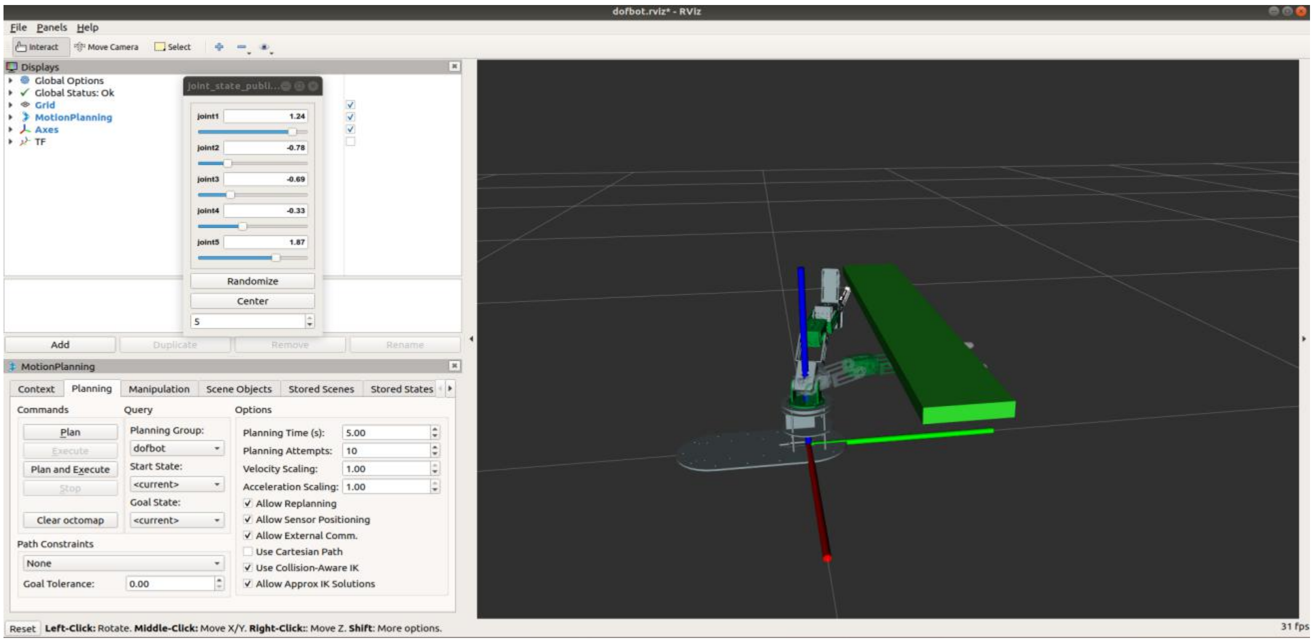
Return to Planning and plan the robotic arm again.It will never collide with the box, and it will never reach the position of the box.

About code:

Add an obstacle with length, width and height [0.7, 0.1, 0.02] in front of the robotic arm, and the center of the obstacle is located at xyz(0,0.15,0.21)

```
# Initialize the scene object
scene = PlanningSceneInterface()
# Set height of desktop
table_ground = 0.2
# Set the three-dimensional size of obstacles [length, width, and height]
table_size = [0.7, 0.1, 0.02]
# Add the table to the scene
table_pose = PoseStamped()
table_pose.header.frame_id = 'base_link'
table_pose.pose.position.x = 0
table_pose.pose.position.y = 0.15
table_pose.pose.position.z = table_ground + table_size[2] / 2.0
table_pose.pose.orientation.w = 1.0
```

Automatic obstacle avoidance show:


dofbot.rviz* - RViz

- Global Options
- Global Status: Ok
- Grid
- MotionPlanning
- Axes
- TF

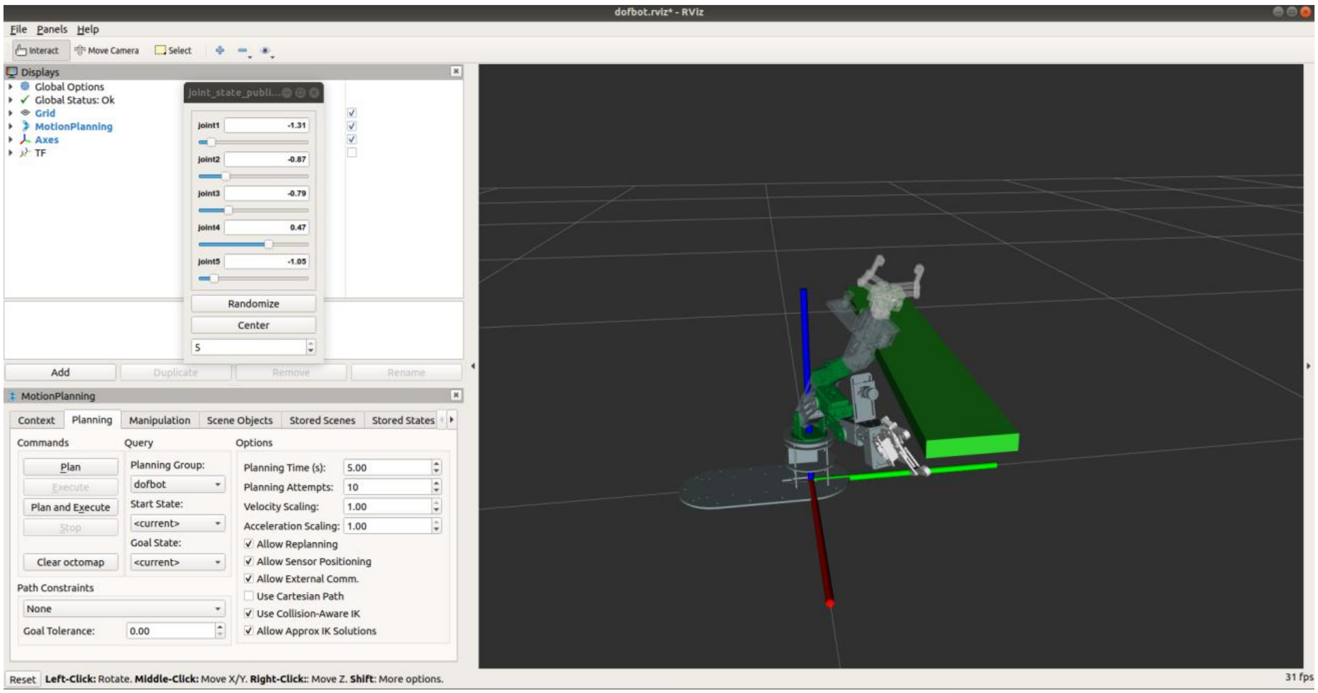
joint_state_publi...

```
joint1   1.24
joint2  -0.78
joint3  -0.69
joint4  -0.33
joint5   1.87
```

Planning Group: dofbot
Start State: <current>
Goal State: <current>
Path Constraints: None
Goal Tolerance: 0.00
Planning Time (s): 5.00
Planning Attempts: 10
Velocity Scaling: 1.00
Acceleration Scaling: 1.00

- ✓ Allow Replanning
- ✓ Allow Sensor Positioning
- ✓ Allow External Comm.
- Use Cartesian Path
- ✓ Use Collision-Aware IK
- ✓ Allow Approx IK Solutions

Reset **Left-Click:** Rotate. **Middle-Click:** Move X/Y. **Right-Click:** Move Z. **Shift:** More options.


dofbot.rviz* - RViz

- Global Options
- Global Status: Ok
- Grid
- MotionPlanning
- Axes
- TF

joint_state_publi...

```
joint1  -1.31
joint2  -0.87
joint3  -0.79
joint4   0.47
joint5  -1.05
```

Planning Group: dofbot
Start State: <current>
Goal State: <current>
Path Constraints: None
Goal Tolerance: 0.00
Planning Time (s): 5.00
Planning Attempts: 10
Velocity Scaling: 1.00
Acceleration Scaling: 1.00

- ✓ Allow Replanning
- ✓ Allow Sensor Positioning
- ✓ Allow External Comm.
- Use Cartesian Path
- ✓ Use Collision-Aware IK
- ✓ Allow Approx IK Solutions

Reset **Left-Click:** Rotate. **Middle-Click:** Move X/Y. **Right-Click:** Move Z. **Shift:** More options.

4. Additional cases

Import the pre-planned scene first..

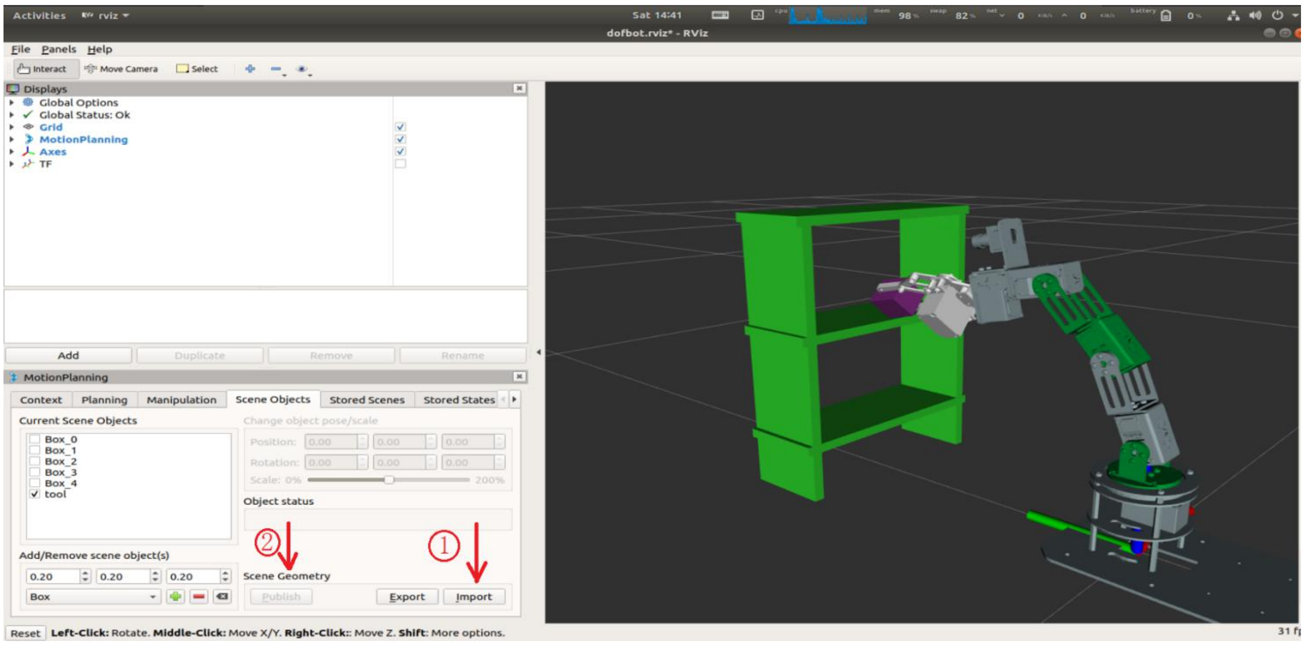
Click **[Import]**, select the scene dofbot_ws/src/dofbot_config/scene/floor.scene; click [Publish] to take effect.

Code path: dofbot_ws/src/dofbot_moveit/scripts/04_Set_Scene.py

Input following command to start this function.

```
cd dofbot_ws/
source devel/setup.bash
rosrun dofbot_moveit 04_Set_Scene.py  # python file
```

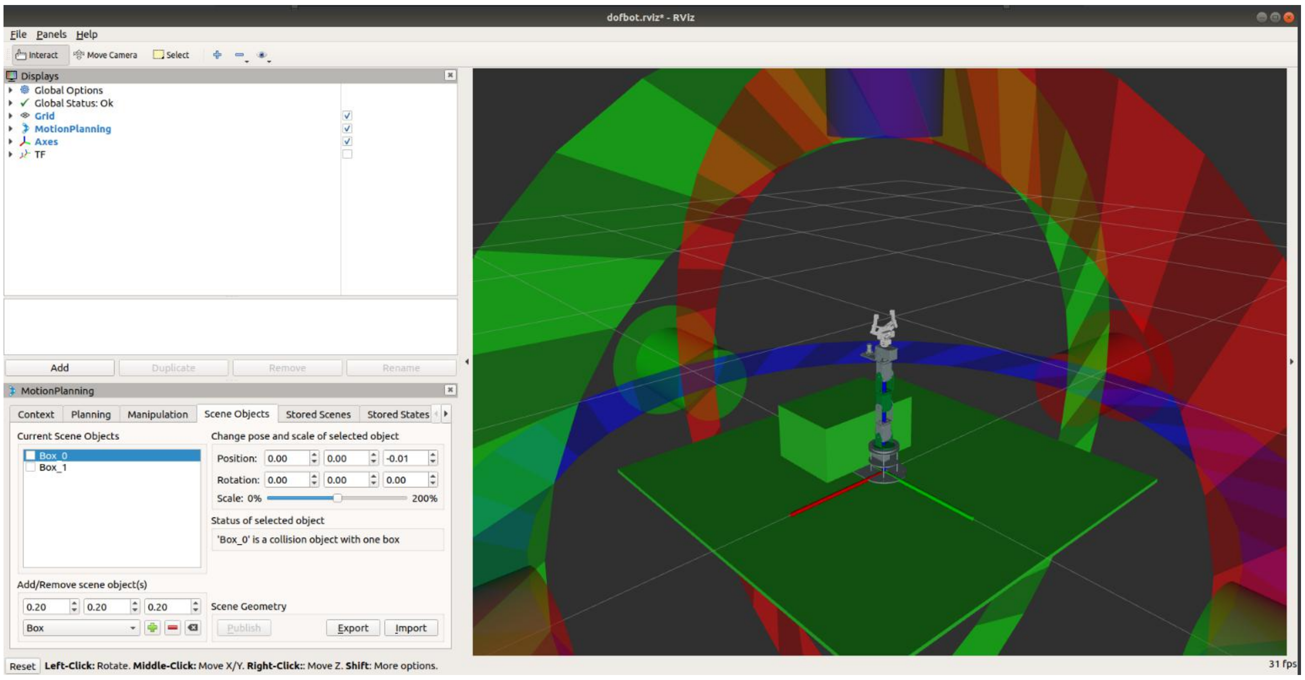
Close case: [ctrl+c] or [ctrl+z]
Click [x] to delete all obstacles.
As shonw below.



5. Drive Actual DOFBOT

Note: Before running the MoveIt case in combination with a real robotic arm, please close the APP remote control process first.

Import the pre-planned scene first to ensure safety.
Click **[Import]**, select the scene dofbot_ws/src/dofbot_config/scene/floor.scene; click [Publish] to take effect.



Current Scene Objects: Box_0, Box_1

Change pose and scale of selected object
Position: 0.00 0.00 -0.01
Rotation: 0.00 0.00 0.00
Scale: 0% 200%

Status of selected object
'Box_0' is a collision object with one box

Add/Remove scene object(s)
0.20 0.20 0.20
Box

Scene Geometry

Left-Click: Rotate. **Middle-Click:** Move X/Y. **Right-Click::** Move Z. **Shift:** More options.

Open the command terminal, input following command:

```
rosropic list
```

Input following command to check the topic, there is a topic of /joint_states.

```
rostopic echo /joint_states
```

If you want to publish current joint angle of robotic arm in real time in the simulation environment. You only need to write a subscriber to subscribe to the topic in real time and drive the robotic arm to move in real time.
Note: Please pay attention to safety when driving the robotic arm.

About code:
Code path: dofbot_ws/src/dofbot_moveit/scripts/00_dofbot_move.py

Create a subscriber

```
subscriber = rospy.Subscriber("/joint_states", JointState, topic)
```

Subscriber callback function

```
# Define the joint angle container, the last one is the angle of the gripper, the default gripper does not move is 0.
if not isinstance(msg, JointState): return
# Define the joint angle container, the last one is the angle of the gripper, the default gripper does not move is 0.
joints = [0.0, 0.0, 0.0, 0.0, 0.0, 0.0]
# Convert the received radian [-1.57,1.57] into an angle [0,180]
```

```
for i in range(5): joints[i] = (msg.position[i] * RA2DE) + 90
```

The gripper does not participate in the positive and negative solution of the movement, it can be controlled separately. Do not set the No. 6 servo to 0°.

Input following command to start this function.

```
python 00_dofbot_move.py
# or
rosrun dofbot_moveit 00_dofbot_move.py
```

Turn off the code, and the robotic arm will stop moving.

If you want to use another device for remote control, ROS multi-machine communication is required.

First, ensure that the two devices have the same Ubuntu environment and the same ROS environment.
View the IP information and host name of the two devices.

```
Command : ifconfig and hostname
```

Modify hosts file in the /etc folder.

```
sudo chmod a+w /etc/hosts
sudo vim /etc/hosts
```

Add the IP and host names of the two devices to their hosts files, and bind users. The first is the IP and the back is the name.
After modification, enter the following commands on both devices to restart the network.

```
sudo /etc/init.d/networking restart
```

Realize communication between two devices.
Install the chrony package on the two devices for synchronization.

```
sudo apt-get install chrony
```

Input following command install ssh server on master and slave device.

```
sudo apt-get install openssh-server
```

After installation, input following command to confirm whether the server has been started.

```
ps -e|grep ssh
```

Modify ～/.bashrc file on master and slave device.
Note: At the end of this function, you need to comment out or restore the modified part of the .bashrc file, otherwise it will cause the APP remote control process to fail to start.

```
sudo vim ~/.bashrc
```

Add this following content to the .bashrc file on the master device.

```
export ROS_HOSTNAME= Master user name
export ROS_MASTER_URI=http://Master user name:11311
```

Add this following content to the .bashrc file on the slave device.

```
export ROS_HOSTNAME= Slave user name
export ROS_MASTER_URI=http://Slave user name:11311
```

After setting the IP, refresh it, and then you can communicate.

```
source ~/.bashrc
```

After the configuration is complete, the slave starts the subscriber code to drive the real machine movement. The master starts the rviz simulation operation and publishes the topic.

After configuration, execute the following commands on the slave device.

```
cd dofbot_ws/
source devel/setup.bash
rosrun dofbot_moveit 00_dofbot_move.py
```

Close case: [ctrl+c] or [ctrl+z]