6、MoveIt avoiding

This lesson takes the MoveIT simulation as an example. If you need to set the synchronization between the real machine and the simulation, please refer to the lesson [02, MoveIt Precautions and Controlling the Real Machine]. ! ! ! be careful! ! !

The effect demonstration is a virtual machine, and other masters are running (related to the performance of the master, depending on the actual situation).

6.1. Start

Start the MoveIT

```
roslaunch transbot_se_moveit_config demo.launch
```

Start collision detection node

```
rosrun transbot_se_moveit_config 05_attached_object       # C++
rosrun transbot_se_moveit_config 05_attached_object.py    # python
```

Set the random motion for about 10 times. Since the target position is random, it can also randomly reach the obstacle. When the target position contacts the obstacle, the planning will fail.

The effect diagram is as follows

6.2. Source code

6.2.1.py file

Set the size and pose of obstacles

```
    # Set the 3D size of the obstacle [length, width and height]
    table_size = [0.2, 0.1, 0.02]
    table_pose = PoseStamped()
    table_pose.header.frame_id = 'dummy'
    table_pose.pose.position.x = 0
    table_pose.pose.position.y = 0.33
    table_pose.pose.position.z = 0.05
    table_pose.pose.orientation.w = 0.7
```

Add Obstacles to Scene

```
    scene.add_box('obj', table_pose, table_size)
```

Set random move

```
    index = 0
    while index < 10:
        # Set random target points
        transbot.set_random_target()
        # Start exercising
        transbot.go()
        sleep(0.5)
        index += 1
        print ("Plan {}th time!!!".format(index))
```

6.2.2.C++ file

Set the size and pose of obstacles

```
    moveit::planning_interface::PlanningSceneInterface scene;
    //////////////////////////Add obstacles/////////////////////
    vector<string> object_ids;
    scene.removeCollisionObjects(object_ids);
    vector<moveit_msgs::CollisionObject> objects;
    moveit_msgs::CollisionObject obj;
    //Set the id of the obstacle
    obj.id = "obj";
    object_ids.push_back(obj.id);
    //Status of obstacles
    obj.operation = obj.ADD;
    obj.header.frame_id = frame;
    shape_msgs::SolidPrimitive primitive;
    //Set the obstacle type
    primitive.type = primitive.BOX;
    //Set obstacle dimension
    primitive.dimensions.resize(3);
    //Set the length, width and height of obstacles
    primitive.dimensions[0] = 0.2;
    primitive.dimensions[1] = 0.1;
    primitive.dimensions[2] = 0.02;
    obj.primitives.push_back(primitive);
    geometry_msgs::Pose pose;
    //Set the position information of the obstacle [x,y,z]
    pose.position.x = 0;
    pose.position.y = 0.33;
    pose.position.z = 0.05;
    tf::Quaternion quaternion;
    //The unit of R,P,Y is angle
    double Roll = 0.0;
    double Pitch = 0.0;
    double Yaw = 0.0;
    quaternion.setRPY(Roll * M_PI / 180, Pitch * M_PI / 180, Yaw * M_PI / 180);
    pose.orientation.x = quaternion.x();
```

```
    pose.orientation.y = quaternion.y();
    pose.orientation.z = quaternion.z();
    pose.orientation.w = quaternion.w();
    //Set the pose information of obstacles
    obj.primitive_poses.push_back(pose);
    objects.push_back(obj);
```

Set the color of obstacles

```
    ////////////////////////Set the color of obstacles/////////////////////
    //Create a color container for detected objects
    std::vector<moveit_msgs::ObjectColor> colors;
    moveit_msgs::ObjectColor color;
    //Add the id that needs to set the color
    color.id = "obj";
    //Set RGBA value, range [0~1]
    color.color.r = 1.0;
    color.color.g = 0;
    color.color.b = 0;
    color.color.a = 0.5;
    colors.push_back(color);
    //Add set information to the scene
    scene.applyCollisionObjects(objects, colors);
```

Set random move

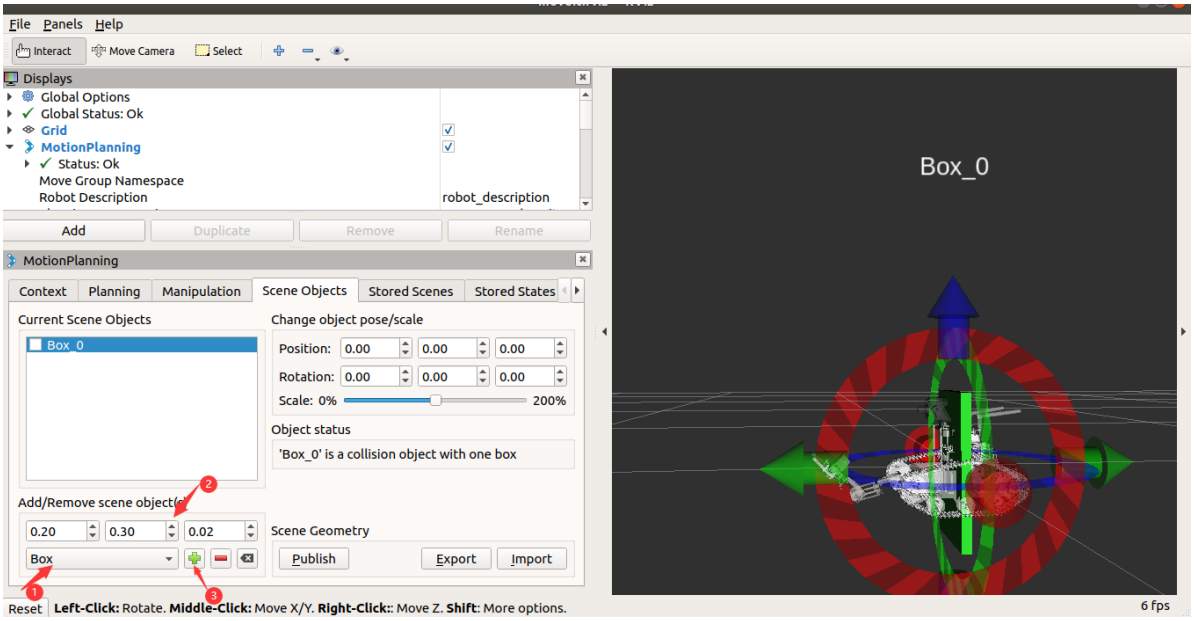
```
    int index = 0;
    while (index < 10) {
        transbot.setRandomTarget();
        transbot.move();
        sleep(0.5);
        index++;
        cout << " the " << index << "planning!!!" << endl;
    }
```

6.3.Scene Object

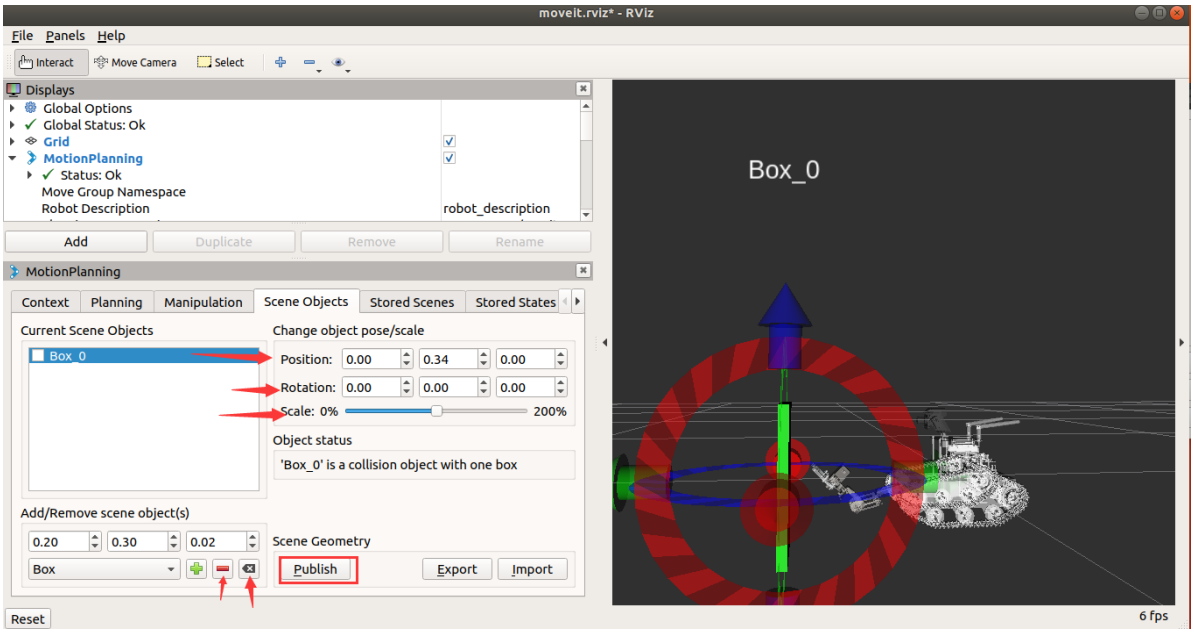
In addition to using code to add or delete obstacles, you can also use the [MotionPlanning] tool in rviz to add or delete them.

6.3.1. Add obstacles

The first step is to select the shape of the added obstacle [for example: Box], then set the size of the obstacle, and then click the [+] sign. At this time, the pose of the obstacle is in the center of the origin. As shown below



The second step, drag the right arrow to drag the obstacle out, or directly set the [Position] position information, [Rotation] attitude information, [Scale] scaling ratio, etc. As shown below



The third step, after completing the above setting steps, be sure to click [Publish], it will take effect; otherwise, the robot may pass through obstacles.

6.3.2.Delete obstacles

As shown above,

Delete: Select the obstacle first, and then click the [-] sign to delete the current obstacle.

Empty: directly click the [x] to clear the obstacles.