Experiment 3

Familiarization with RVIZ

Objectives

- To familiarize with the tool RVIZ
- Display basic shapes in RVIZ using ROS
- Display points and lines in RVIZ using ROS

Theory

The RViz tool is an official 3D visualization tool of ROS. Almost all kinds of data from sensors can be viewed through this tool. RViz will be installed along with the ROS desktop full installation

Procedure

The experiment is divided into two parts.

1. Sending basic shapes

1. Create a new package for visualization using following command
   >> catkin_create_pkg using_markers roscpp visualization_msgs
2. Inside the using_markers/src create a new file called basic_shapes.cpp and copy the code

```
/*
*  Copyright (c) 2010, Willow Garage, Inc.
*  All rights reserved.
 *
*  Redistribution and use in source and binary forms, with or without  * modification, are permitted provided that the following conditions are met:
 *
*  * Redistributions of source code must retain the above copyright
*  notice, this list of conditions and the following disclaimer.
*  * Redistributions in binary form must reproduce the above copyright  *  notice, this list of conditions and the following disclaimer in the
*  documentation and/or other materials provided with the distribution.
*  * Neither the name of the Willow Garage, Inc. nor the names of its  *  contributors may be used to endorse or promote products derived from
*  this software without specific prior written permission.
 *
*  THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS"
*  AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
*  IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
*  ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR CONTRIBUTORS BE
*  LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR
*  CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF
*  SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS
*  INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN
*  CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
*  ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
```

```
*                POSSIBILITY OF SUCH DAMAGE.
 */

// %Tag(FULLTEXT)%
// %Tag(INCLUDES)%
#include <ros/ros.h>
#include <visualization_msgs/Marker.h>
// %EndTag(INCLUDES)%

// %Tag(INIT)%
int main( int argc, char** argv )
{
  ros::init(argc, argv, "basic_shapes");   ros::NodeHandle n;
ros::Rate r(1);   ros::Publisher marker_pub =
n.advertise<visualization_msgs::Marker>("visualization_marker", 1);
// %EndTag(INIT)%

  // Set our initial shape type to be a cube
// %Tag(SHAPE_INIT)%
  uint32_t shape = visualization_msgs::Marker::CUBE;
// %EndTag(SHAPE_INIT)%

// %Tag(MARKER_INIT)%   while
(ros::ok())
  {
    visualization_msgs::Marker marker;     // Set the frame ID and timestamp.  See the TF
tutorials for information on these.
    marker.header.frame_id = "/my_frame";     marker.header.stamp = ros::Time::now();
// %EndTag(MARKER_INIT)%

    // Set the namespace and id for this marker.  This serves to create a unique ID
    // Any marker sent with the same namespace and id will overwrite the old one // %Tag(NS_ID)%
    marker.ns = "basic_shapes";     marker.id =
0; // %EndTag(NS_ID)%

    // Set the marker type.  Initially this is CUBE, and cycles between that and SPHERE, ARROW, and
CYLINDER
// %Tag(TYPE)%     marker.type =
shape;
// %EndTag(TYPE)%

    // Set the marker action.  Options are ADD, DELETE, and new in ROS Indigo: 3 (DELETEALL) //
%Tag(ACTION)%
    marker.action = visualization_msgs::Marker::ADD;
// %EndTag(ACTION)%

    // Set the pose of the marker.  This is a full 6DOF pose relative to the frame/time specified in the header
// %Tag(POSE)%
    marker.pose.position.x = 0;     marker.pose.position.y = 0;
marker.pose.position.z = 0;     marker.pose.orientation.x = 0.0;
marker.pose.orientation.y = 0.0;     marker.pose.orientation.z = 0.0;
marker.pose.orientation.w = 1.0;
// %EndTag(POSE)%

    // Set the scale of the marker -- 1x1x1 here means 1m on a side
```

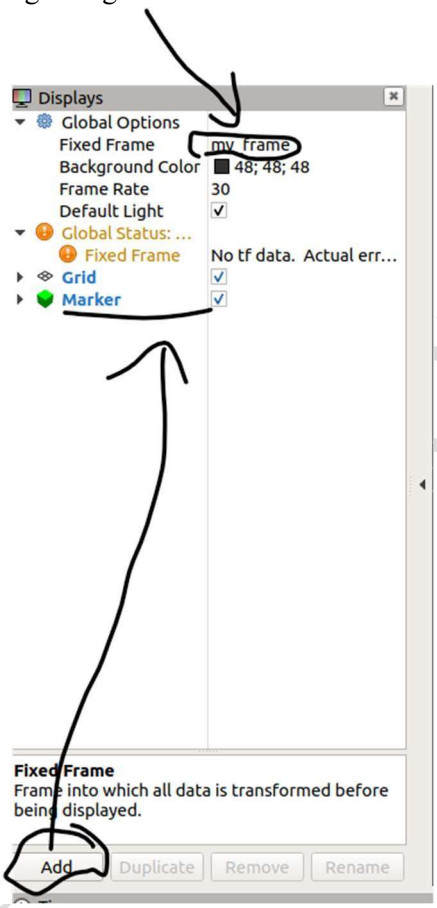
```
// %Tag(SCALE)%     marker.scale.x = 1.0;
marker.scale.y = 1.0;     marker.scale.z =
1.0;
// %EndTag(SCALE)%

   // Set the color -- be sure to set alpha to something non-zero!
// %Tag(COLOR)%     marker.color.r = 0.0f;
marker.color.g = 1.0f;     marker.color.b =
0.0f;     marker.color.a = 1.0;
// %EndTag(COLOR)%

// %Tag(LIFETIME)%
   marker.lifetime = ros::Duration();
// %EndTag(LIFETIME)%

   // Publish the marker //
%Tag(PUBLISH)%
   while (marker_pub.getNumSubscribers() < 1)
   {
    if (!ros::ok())
    {        return 0;
    }
    ROS_WARN_ONCE("Please create a subscriber to the marker");      sleep(1);     }
   marker_pub.publish(marker);
// %EndTag(PUBLISH)%

   // Cycle between different shapes
// %Tag(CYCLE_SHAPES)%     switch
(shape)
   {     case visualization_msgs::Marker::CUBE:
     shape = visualization_msgs::Marker::SPHERE;      break;     case
visualization_msgs::Marker::SPHERE:       shape =
visualization_msgs::Marker::ARROW;       break;     case
visualization_msgs::Marker::ARROW:
     shape = visualization_msgs::Marker::CYLINDER;       break;     case
visualization_msgs::Marker::CYLINDER:       shape =
visualization_msgs::Marker::CUBE;       break;
   }
// %EndTag(CYCLE_SHAPES)%

// %Tag(SLEEP_END)%
   r.sleep();
 }
// %EndTag(SLEEP_END)%
}
// %EndTag(FULLTEXT)%
```

3. Now edit the CMakeLists.txt file in your using_markers package, and add following lines to bottom:

   add_executable(basic_shapes src/basic_shapes.cpp) target_link_libraries(basic_shapes ${catkin_LIBRARIES})
4. Come to catkin_ws and enter the code >>catkin_make
5. Run the following command to build rviz
   >>rosmake rviz
6. Begin a new master using the code

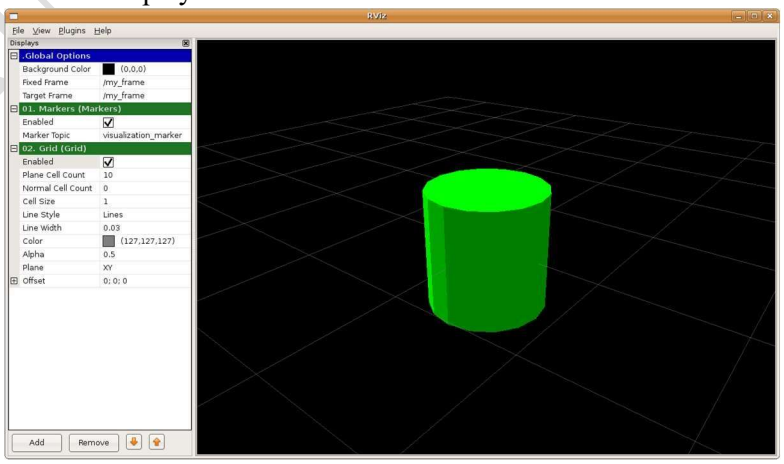
>> roscore

7. Run rviz using the command in a new tab
   >> rosrun rviz rviz
8. Run the following code in new terminal
   >>rosrun using_markers basic_shapes
9. Open RVIZ and make following changes



The entry in the right of fixed frame is 'map' by default. Change to /my_frame

10. Next add a Markers display. Notice that the default topic specified, visualization_marker, is the same as the one being published.
11. The output will be displayed in the screen



II. Markers: Points and Lines

1. In the using_markers/src create a new file titled points_and_lines.cpp
2. Copy the code to the file

```
/*
* Copyright (c) 2010, Willow Garage, Inc.
* All rights reserved.
*
* Redistribution and use in source and binary forms, with or without
* modification, are permitted provided that the following conditions are met:
*
*     * Redistributions of source code must retain the above copyright
*       notice, this list of conditions and the following disclaimer.
*     * Redistributions in binary form must reproduce the above copyright
*       notice, this list of conditions and the following disclaimer in the
*       documentation and/or other materials provided with the distribution.
*     * Neither the name of the Willow Garage, Inc. nor the names of its
*       contributors may be used to endorse or promote products derived from
*       this software without specific prior written permission.
*
* THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS"
* AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
* IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
* ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR CONTRIBUTORS BE
* LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR
* CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF
* SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS
* INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN
* CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
* ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
* POSSIBILITY OF SUCH DAMAGE. 30.  */

// %Tag(FULLTEXT)%
#include <ros/ros.h>
#include <visualization_msgs/Marker.h>

#include <cmath>

  int main( int argc, char** argv )
  {
  ros::init(argc, argv, "points_and_lines");
  ros::NodeHandle n;
  ros::Publisher marker_pub =
                              n.advertise<visualization_msgs::Marker>("visualization_marker", 10);

ros::Rate r(30);
```

```
45.
46.         float f = 0.0;
47.         while (ros::ok())
48.         {
49.         // %Tag(MARKER_INIT)%
50.         visualization_msgs::Marker points, line_strip, line_list;
51.         points.header.frame_id = line_strip.header.frame_id = line_list.header.frame_id =
            "/my_frame";
52.         points.header.stamp = line_strip.header.stamp = line_list.header.stamp = ros::Time::now();
53.         points.ns = line_strip.ns = line_list.ns = "points_and_lines";
54.         points.action = line_strip.action = line_list.action = visualization_msgs::Marker::ADD;
55.         points.pose.orientation.w = line_strip.pose.orientation.w = line_list.pose.orientation.w = 1.0;
            56. // %EndTag(MARKER_INIT)%
57.
58.         // %Tag(ID)%
59.         points.id = 0;
60.         line_strip.id = 1;
61.         line_list.id = 2; 62. // %EndTag(ID)%
63.
64.         // %Tag(TYPE)%
65.         points.type = visualization_msgs::Marker::POINTS;
66.         line_strip.type = visualization_msgs::Marker::LINE_STRIP;
67.         line_list.type = visualization_msgs::Marker::LINE_LIST; 68. // %EndTag(TYPE)%
69.
70.         // %Tag(SCALE)%
71.         // POINTS markers use x and y scale for width/height respectively
72.         points.scale.x = 0.2;
73.         points.scale.y = 0.2;
74.
75.         // LINE_STRIP/LINE_LIST markers use only the x component of scale, for the line width
76.         line_strip.scale.x = 0.1;
77.         line_list.scale.x = 0.1;
78.         // %EndTag(SCALE)%
79.
80.         // %Tag(COLOR)%
81.         // Points are green
82.         points.color.g = 1.0f; 83.     points.color.a = 1.0;
84.
85.         // Line strip is blue
86.         line_strip.color.b = 1.0;
87.         line_strip.color.a = 1.0;
88.
89.         // Line list is red
90.         line_list.color.r = 1.0;
91.         line_list.color.a = 1.0; 92. // %EndTag(COLOR)%
93.
94.             // %Tag(HELIX)%
95.             // Create the vertices for the points and lines
96.             for (uint32_t i = 0; i < 100; ++i)
97.             {
98.             float y = 5 * sin(f + i / 100.0f * 2 * M_PI);
99.             float z = 5 * cos(f + i / 100.0f * 2 * M_PI); 100.
101.            geometry_msgs::Point p;
102.            p.x = (int32_t)i - 50;
103.            p.y = y;
```

```
104.                p.z = z; 105.
106.      points.points.push_back(p); 107.
line_strip.points.push_back(p); 108.
109.                // The line list needs two points for each line
110.                line_list.points.push_back(p);
111.                p.z += 1.0;
112.                line_list.points.push_back(p);
113.                }
114.                // %EndTag(HELIX)% 115.
116.            marker_pub.publish(points);
117.            marker_pub.publish(line_strip); 118.     marker_pub.publish(line_list);
                119.
120.    r.sleep(); 121.
122.            f += 0.04;
123.            }
124.            }
125.            // %EndTag(FULLTEXT)%
```

3. Edit the CMakeLists.txt file in your using_markers package, and add to the bottom:

```
add_executable(points_and_lines src/points_and_lines.cpp)
target_link_libraries(points_and_lines ${catkin_LIBRARIES})
```

4. Run >> catkin_make
5. Run>>roscore
6. In new tab after setting up the bash run >>rosrun rviz rviz
7.In new tab after setting up the bash run >>rosrun using_markers points_and_lines