src/bot/src/CPose.cpp

```
#include "CPose.h"

// Implementation file for class CLidar
// Functions :
//            - Constructor
//            - Destructor
//            - Call back function sub to odom msg
//            - Pose Publshing function
//            - Trajectory plotting function

//---Constructor
CPose::CPose():nh_priv_("~")
{
  ROS_INFO("Pose node initalised");
  // Subscribe to odometry topic
  odomSub = nh_.subscribe("odom", QSize, &CPose::odomMsgCallBack, this);

  //ROS publisher to publish to a new topic
  botPub = nh_.advertise<std_msgs::Float64>(topicName,QSize);
  TrajectoryPub = nh_.advertise<visualization_msgs::Marker>(trajectoryTopic,
QSize);

  // Pose data from odometry
  tb3Pose= 0.0;
  ROS_ASSERT(true);
}

//--- Destructor
CPose::~CPose()
{
  trajectoryMsg.points.clear();
  ros::shutdown;
}

//--- Call back function sub to odom msg
void CPose::odomMsgCallBack(const nav_msgs::Odometry::ConstPtr &msg)
{
  // Compute current odometry
  double siny = 2.0 * (msg->pose.pose.orientation.w * msg->
pose.pose.orientation.z + msg->pose.pose.orientation.x * msg->
pose.pose.orientation.y);
    double cosy = 1.0 - 2.0 * (msg->pose.pose.orientation.y * msg->
pose.pose.orientation.y + msg->pose.pose.orientation.z * msg->
pose.pose.orientation.z);

    tb3Pose = atan2(siny, cosy);
  odomPose = msg->pose.pose;
}

//---Publshing function
void CPose::PublishPose()
{
  msg.data = tb3Pose;
  botPub.publish(msg);
}

//--- Function publishes visualisation markers for path plot
```

```
53 void CPose::TrajectoryVisualise()
54 {
55   // TRAJECTORY MSGS
56   //http://wiki.ros.org/rviz/Tutorials/Markers%3A%20Basic%20Shapes
57   // Modify trajectory msg for publhsing
58   trajectoryMsg.header.frame_id = "map";            // which frame to use
59   trajectoryMsg.header.stamp = ros::Time();         // timing
60   trajectoryMsg.frame_locked = true;                // Lock frame
61
62   // set id and namespace for rViz to access points to plot
63   trajectoryMsg.ns = "points";
64   trajectoryMsg.id = 0;
65
66   // Set shape type of points and  tell msgs to add points
67   // on rviz
68   trajectoryMsg.type = visualization_msgs::Marker::POINTS;
69   trajectoryMsg.action = visualization_msgs::Marker::ADD;
70
71   // Set initial position of trajectory on the map
72   trajectoryMsg.pose.position.x = 0.0;
73   trajectoryMsg.pose.position.y = 0.0;
74   trajectoryMsg.pose.position.z = 0.0;
75
76   // Set initialorienation of trajectory on the map
77   trajectoryMsg.pose.orientation.x = 0.0;
78   trajectoryMsg.pose.orientation.y = 0.0;
79   trajectoryMsg.pose.orientation.z = 0.0;
80   trajectoryMsg.pose.orientation.w = 1.0;
81
82   // Size of plot points
83   trajectoryMsg.scale.x = 0.05;
84   trajectoryMsg.scale.y = 0.05;
85   trajectoryMsg.scale.z = 0.05;
86
87   // Set color of plot  - BLUE
88   trajectoryMsg.color.r = 0.0f;
89   trajectoryMsg.color.g = 0.0f;
90   trajectoryMsg.color.b = 1.0f;
91   trajectoryMsg.color.a = 1.0;
92
93   // Mesh resoruce
94   trajectoryMsg.mesh_resource = "package://pr2_description/meshes/base_v0
   /base.dae";
95
96   // Let points plotted exist on the map as long as simulation is running
97   trajectoryMsg.lifetime = ros::Duration();
98
99   // store list of odom pose.position
100  trajectoryMsg.points.push_back(odomPose.position);
101  TrajectoryPub.publish(trajectoryMsg);
102 }
103 //------------------------------------------------------------
104 // CPose NODE
105 int main(int argc, char* argv[])
106 {
107   ros::init(argc, argv, "Pose_Node");
108   CPose bot;
109   ros::Rate loop_rate(500);
110
```

```
111   while(ros::ok)
112   {
113     bot.PublishPose();
114     bot.TrajectoryVisualise();
115     // process callback for this node
116     ros::spinOnce();
117     loop_rate.sleep();
118   }
119
120   return 0;
121 }
```