% \* Initialize ROS. Connect to the TurtleBot by using gazebo ip address  
rosshutdown;  
clc;clear  
ipaddress='172.29.64.201';  
rosinit(ipaddress)  
gazebo = ExampleHelperGazeboCommunicator();  
% create ball giving parameters like size, color, force   
ball = ExampleHelperGazeboModel('Ball');  
spherelink = addLink(ball,'sphere',0.2,'color',[0 0 1 1]);  
spawnModel(gazebo,ball,[0.5,2,0.5]);  
spawnedBall = ExampleHelperGazeboSpawnedModel(ball.Name,gazebo);  
duration = 2; % Seconds  
forcevec = [1 0 0]; % Newtons  
applyForce(spawnedBall, spherelink, duration, forcevec);  
pause(5);  
% turtlebot base moves towards the spawned model in gazebo  
kobuki = ExampleHelperGazeboSpawnedModel('mobile\_base',gazebo);  
setState(kobuki,'orientation',[0 0 pi/2]);  
% creating gazebo environment  
barrier = ExampleHelperGazeboModel('jersey\_barrier','gazeboDB');  
spawnModel(gazebo,barrier,[1.5,-3,0]); % Right barrier  
pause(1);  
spawnModel(gazebo,barrier,[1.5,6,0]); % Left barrierng gazebo world  
pause(1);  
  
% Create subscribers for the color camera, the cliff sensor, and the bumper sensor.   
% Create publishers for emitting sound and for controlling the robot velocity.  
handles.colorImgSub = exampleHelperTurtleBotEnableColorCamera;  
handles.cliffSub = rossubscriber('/mobile\_base/events/cliff', 'BufferSize', 5);  
handles.bumpSub = rossubscriber('/mobile\_base/sensors/bumper\_pointcloud', 'BufferSize', 5);  
handles.soundPub = rospublisher('/mobile\_base/commands/sound', 'kobuki\_msgs/Sound');  
handles.velPub = rospublisher('/mobile\_base/commands/velocity');  
BlueBallParams.blueMax = 30; % Maximum permissible deviation from pure blue  
BlueBallParams.darkMin = 90; % Minimum acceptable darkness value  
% to get latest real world image seen by kinect  
  latestImg = readImage(handles.colorImgSub.LatestMessage);  
  [c,~,ball] = exampleHelperTurtleBotFindBlueBall(latestImg,BlueBallParams);  
  exampleHelperTurtleBotPlotObject(latestImg,ball,c);  
  pause(3);  
  handles.params = BlueBallParams;  
gains.lin.pgain = 1/100;   
gains.ang.dgain = 1/100;  
  gains.lin = struct('pgain',1/100,'dgain',1/100,'igain',0,'maxwindup',0','setpoint',0.65);  
  gains.ang = struct('pgain',1/400,'dgain',1/500,'igain',0,'maxwindup',0','setpoint',0.5);  
      
 handles.gains = gains;  
 timer1 = timer('TimerFcn',{@exampleHelperTurtleBotTrackingTimer,handles},'Period',0.1,'ExecutionMode','fixedSpacing');  
 timer1.StopFcn = {@exampleHelperTurtleBotStopCallback};  
 start(timer1);  
  pause(1);  
    
   if ~exampleHelperTurtleBotIsPhysicalRobot  
    duration = 2;  
    forceVector = [0 4 0];  
    applyForce(spawnedBall,spherelink,duration,forceVector)  
   end  
  pause(50);         % Continue object tracking for given time  
stop(timer1);  
delete(timerfindall)  
    
return