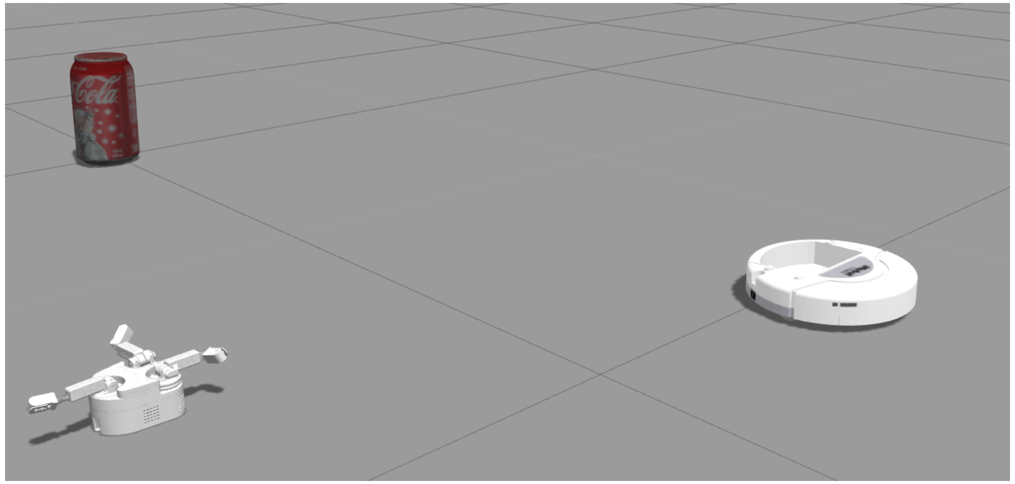
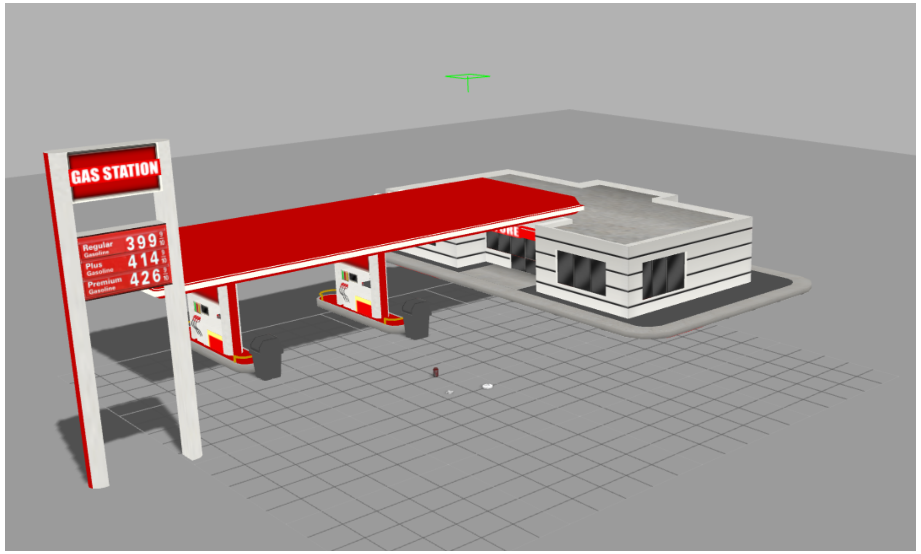
EE-565-Lab2

Introduction to Simulation Environment



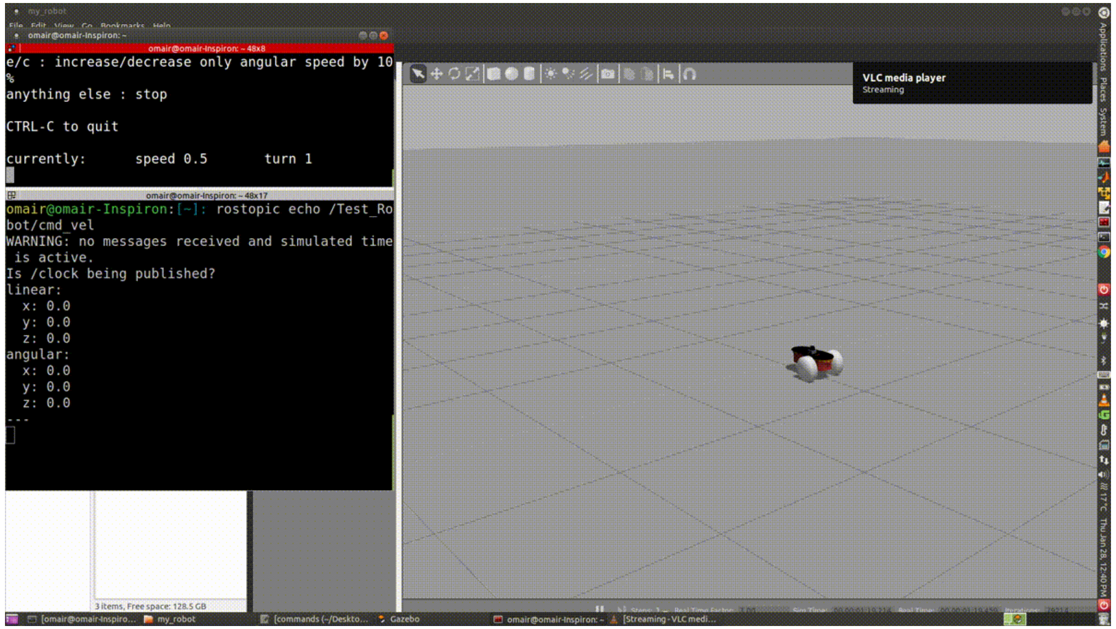
GAZEBO



Dr. Ahmad Kamal Nasir

Today’s Objectives

- Introduction to Gazebo
- Building a robot model in Gazebo
- Populating robot environment with simulated objects
- Writing plugins
- Sensors
- Interface with ROS



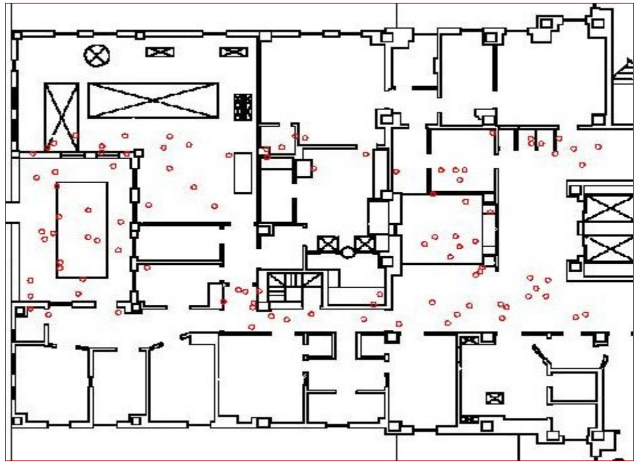
Importance of Simulation

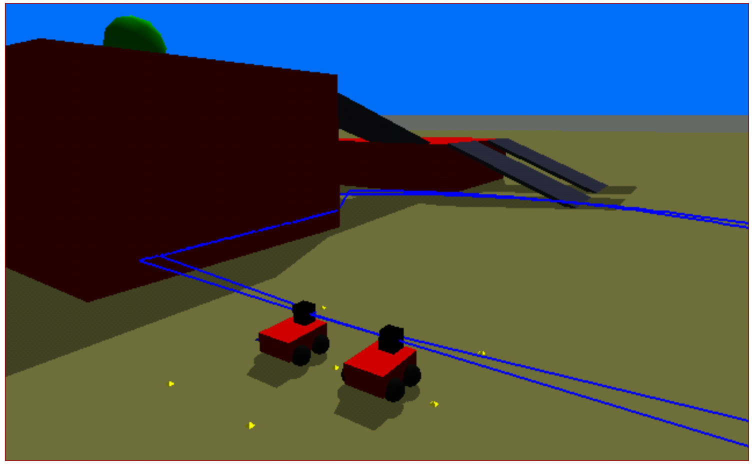
- What
  - Mimic the real world, to a certain extent
- When
  - Always!!
- Why
  - Save time and your sanity
  - Experimentation much less destructive
  - Hardware is much expensive and error prone
  - Simulated Sensors are readily available
  - Create really cool videos
- How
  - Someone has probably already done it, so use it

Which Simulator?

Stage
- 2D
- Sensor-based
- Player interface
- Kinematic
- $O(1) \sim O(n)$
- Large teams (100's)

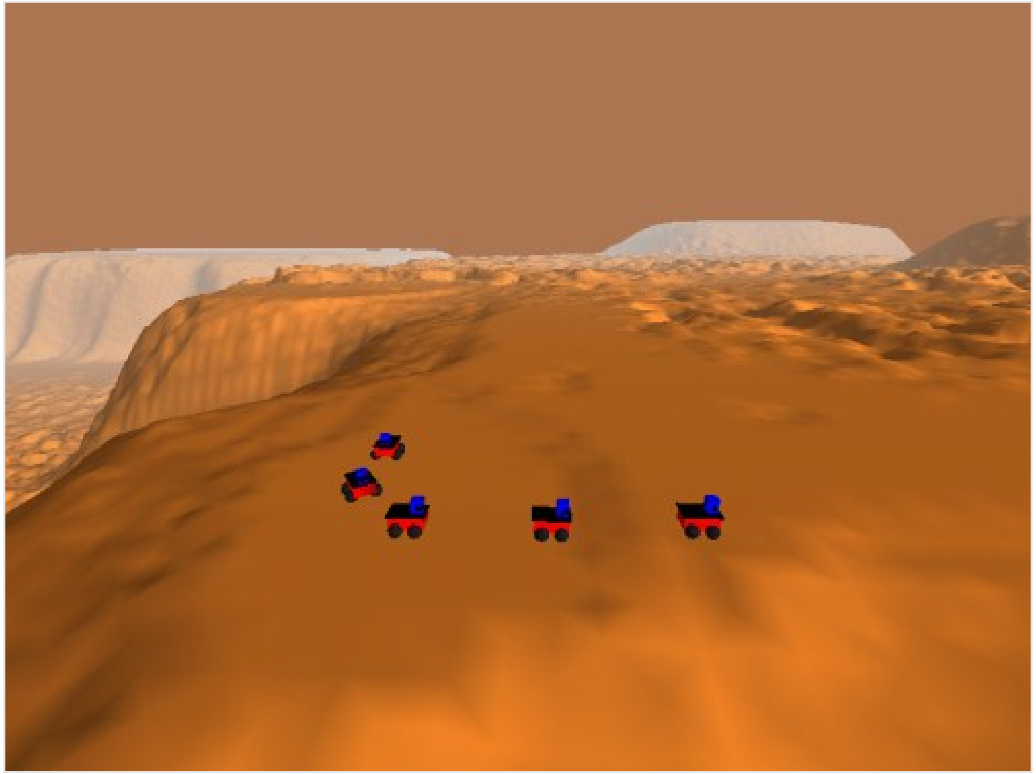
Gazebo
- 3D
- Sensor-based
- Player
- Dynamic
- $O(n) \sim O(n^3)$
- Small teams (10's)





Gazebo

- Simulates robots, sensors, and objects in a 3-D dynamic environment
- Generates realistic sensor feedback and physical interactions between objects



Gazebo (Cont.)

- gzserver
  - executable runs the physics update-loop and sensor data generation
  - This is core of Gazebo, and can be used independently of any graphical interface
- Gzclient
  - executable runs the QT based user interface
  - provides a nice visualization of simulation, and convenient controls over various simulation properties

Gazebo Components

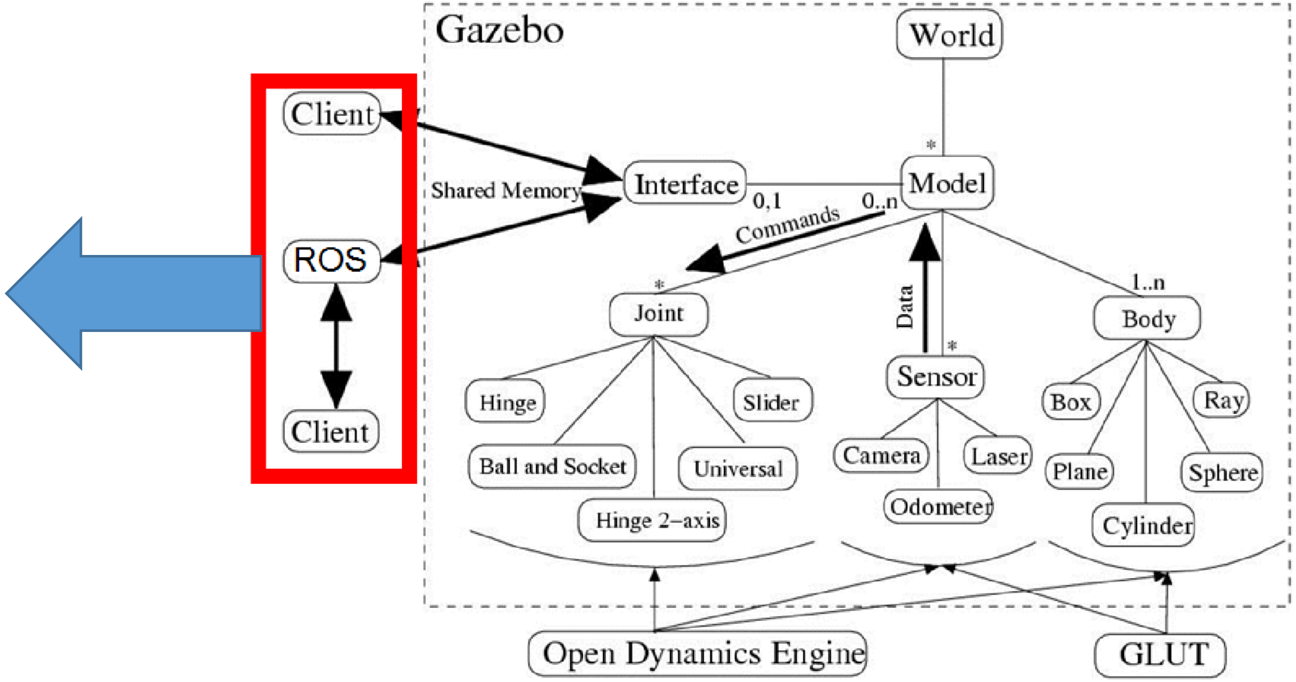
- World File:
  - Contains all the elements in a simulation, including robots, lights, sensors, and static objects. This file is formatted using SDF (Simulation Description Format), and typically has a .world extension
- Model File:
  - A SDF file used to describe a single model.
- Environment Variables:
  - For storing environment, communication settings
- Gazebo Server + Client:
  - The two main components of a simulation
- Plugins:
  - A simple mechanism to interface with the simulation world.

Gazebo Architecture

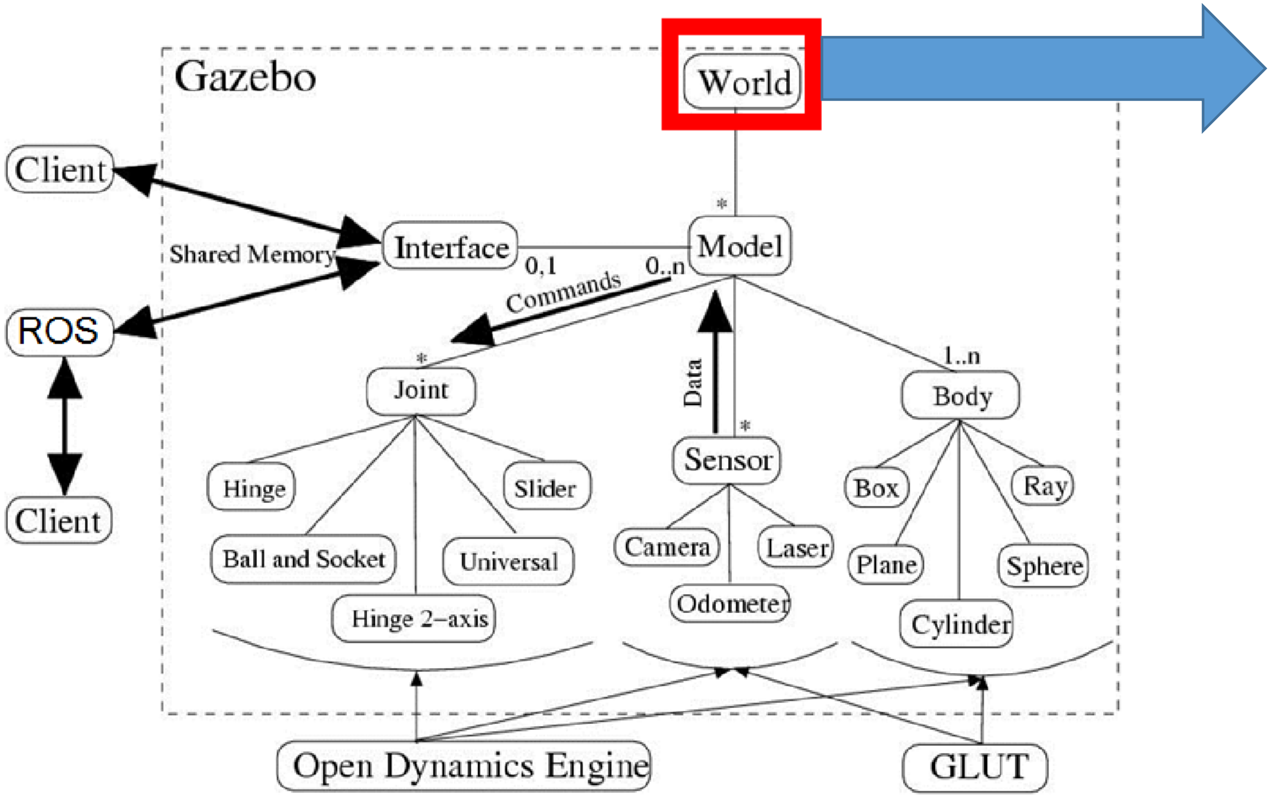
Client code (your program), can interface to Gazebo in two ways

- Libgazebo
  - Shared Memory, direct interface
  - Fast, but requires more work
- ROS
  - Simulation transparency
  - Get all of ROS's goodies
  - Recommended for most cases
  - Gazebo was part of ROS

```
    <link name="left_wheel">
  <pose>0.1 0.13 0.1 0 1.5707 1.5707</pose>
  <collision name="collision">
    <geometry>
      <cylinder>
        <radius>.1</radius>
        <length>.05</length>
      </cylinder>
    </geometry>
  </collision>
  <visual name="visual">
    <geometry>
      <cylinder>
        <radius>.1</radius>
        <length>.05</length>
      </cylinder>
    </geometry>
  </visual>
</link>


<link name="right_wheel">
  <pose>0.1 -0.13 0.1 0 1.5707 1.5707</pose>
```



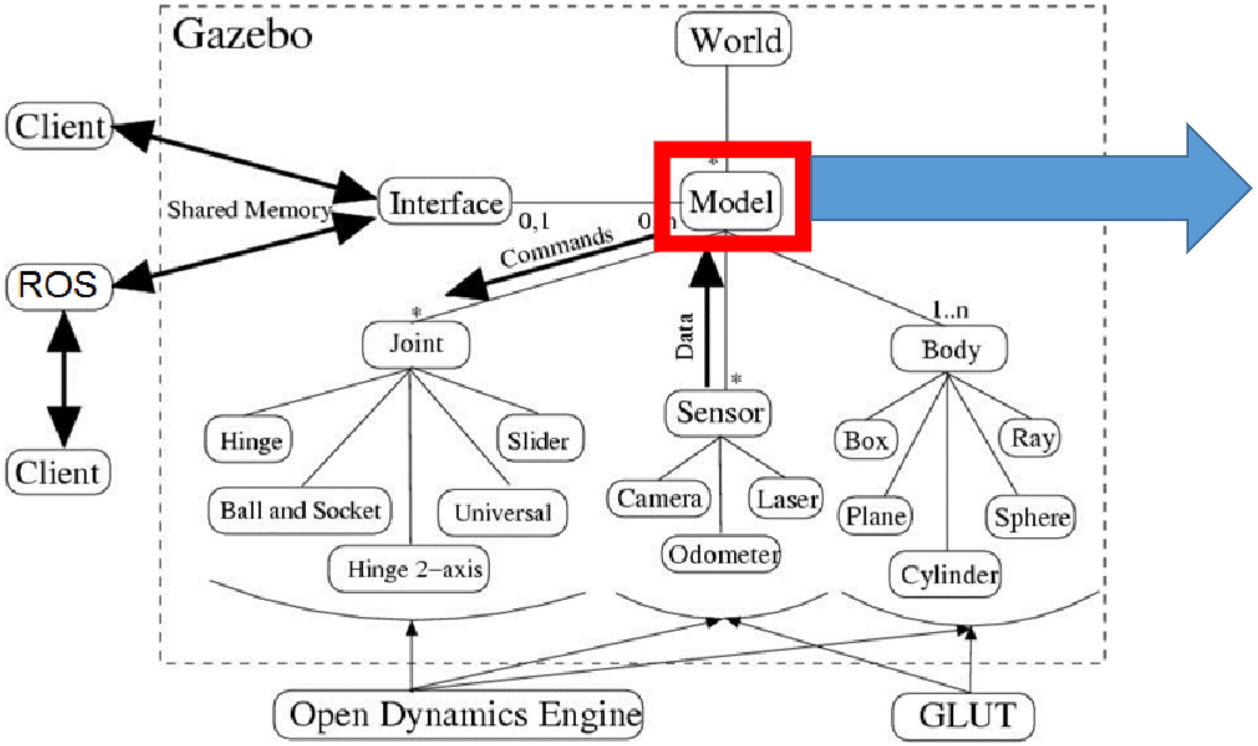
Gazebo Architecture



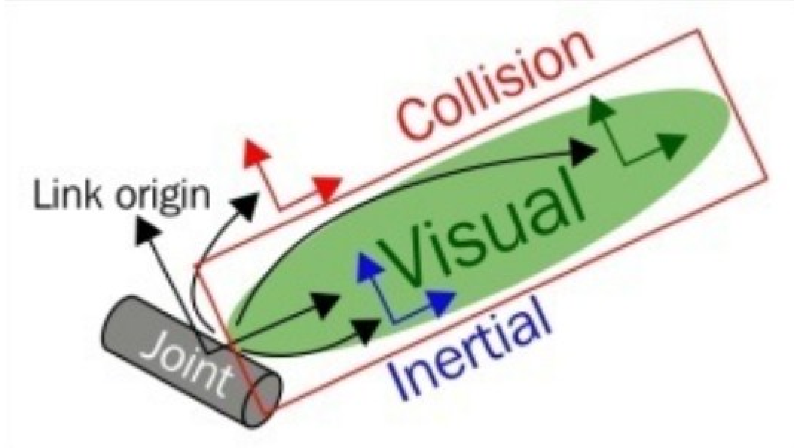
- A world is composed of a model hierarchy
- Models define simulated devices
- Models can be nested
  - Specifies how models are physically attached
  - to one another
  - Think of it as “bolting” one model to another

```
<sdf version='1.4'>

  <world name='default'>
    <light name='sun' type='directional'>
      <cast_shadows>1</cast_shadows>
      <pose>0 0 10 0 -0 0</pose>
      <diffuse>0.8 0.8 0.8 1</diffuse>
      <specular>0.2 0.2 0.2 1</specular>
      <attenuation>
        <range>1000</range>
        <constant>0.9</constant>
        <linear>0.01</linear>
        <quadratic>0.001</quadratic>
      </attenuation>
      <direction>-0.5 0.1 -0.9</direction>
    </light>

    <model name='ground_plane'>
      <static>1</static>
      <link name='link'>
        <collision name='collision'>
          <geometry>
            <plane>
              <normal>0 0 1</normal>
              <size>100 100</size>
            </plane>
          </geometry>
          <surface>
            <friction>
```

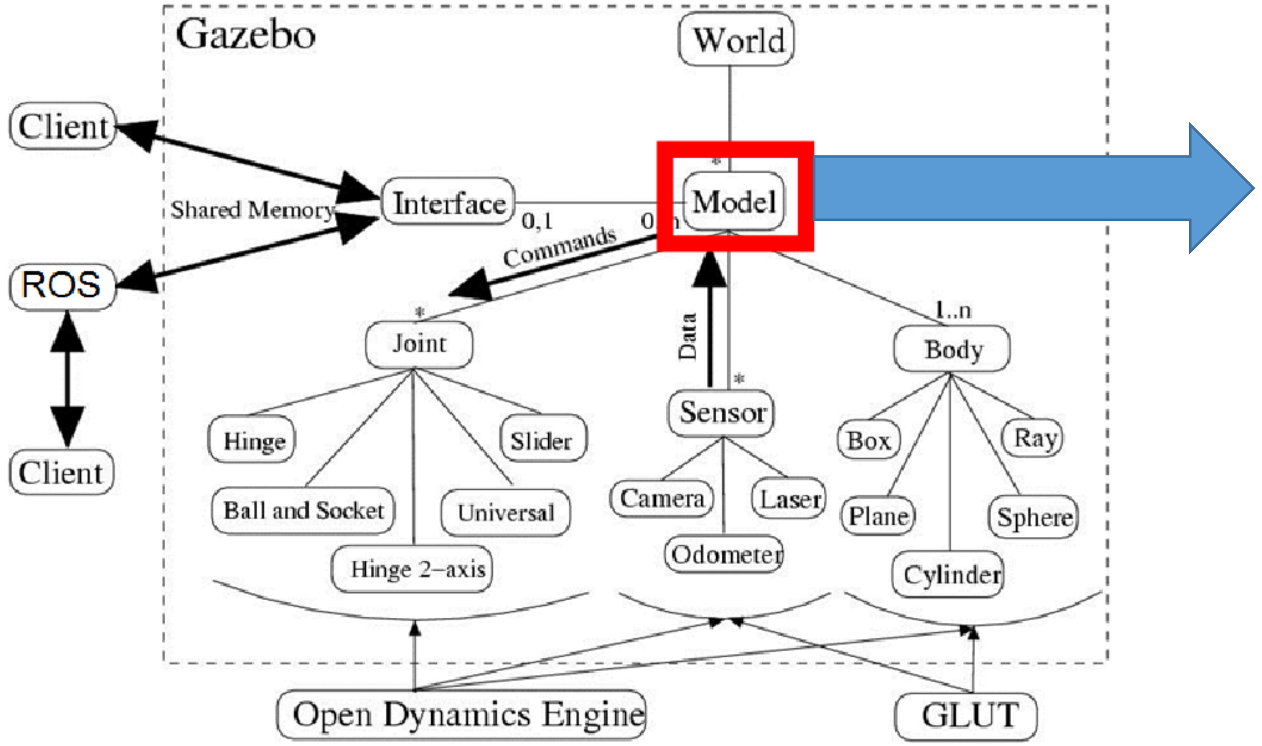
Gazebo Architecture



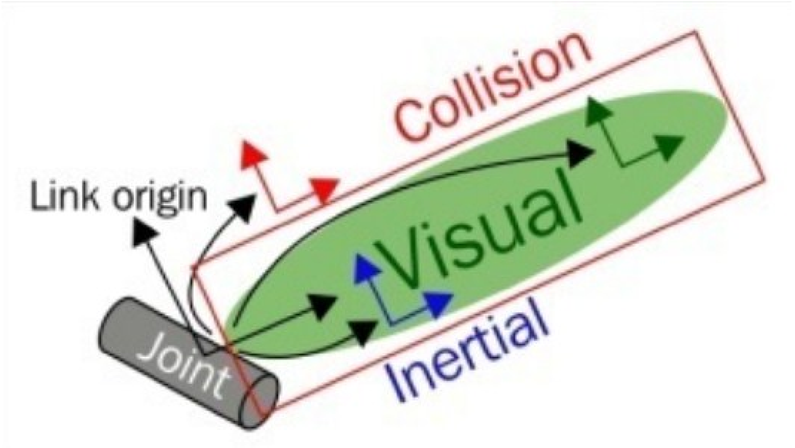
- Each model contains few key properties
  - Physical Presence (optional):
    - Body: sphere, box, composite shapes
    - Kinematics: joints, velocities
    - Dynamics: Mass, friction, forces
    - Appearance: color, texture
  - Interface (optional):
    - Control and feedback interface (libgazebo)



Gazebo Architecture



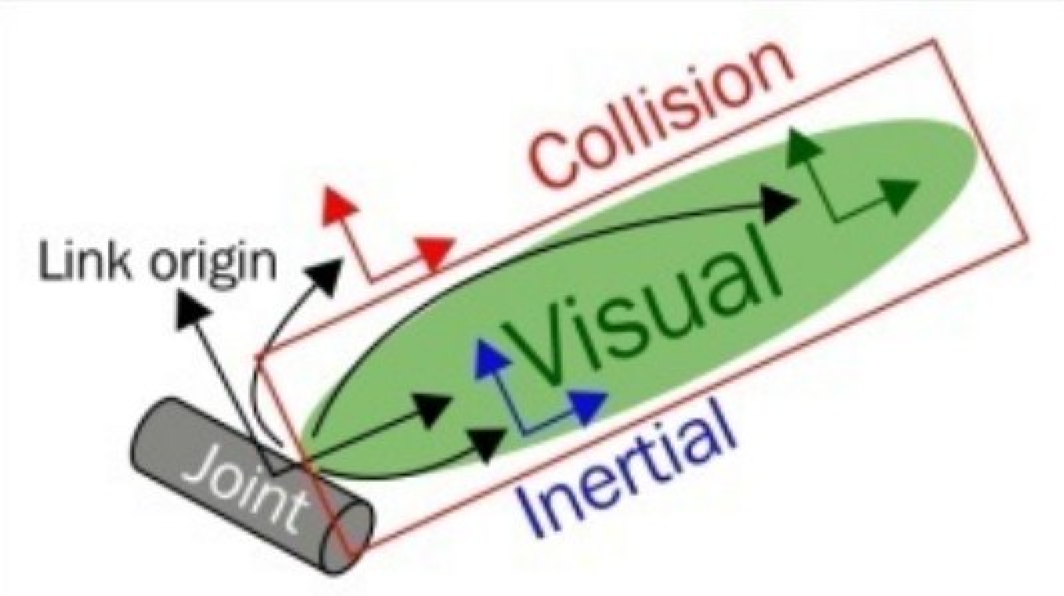
- **Links**: an object may consist of multiple links and can define following properties, e.g. wheel
  - **Visual**: For visualization
  - **Collision**: Encapsulate geometry for collision checking
  - **Inertial:** Dynamic properties of a link e.g. mass, inertia
  - **Sensors:** To collect data from world for plugins
- **Joints:** connect links using a parent-child relationship
- **Plugins:** Library to control model.



Some URDF Tags

- **Link:**
  - Represents single link of robot
  - Includes size, shape, color
  - **Visual** represents real link
  - **Collision** is used to detect collision.

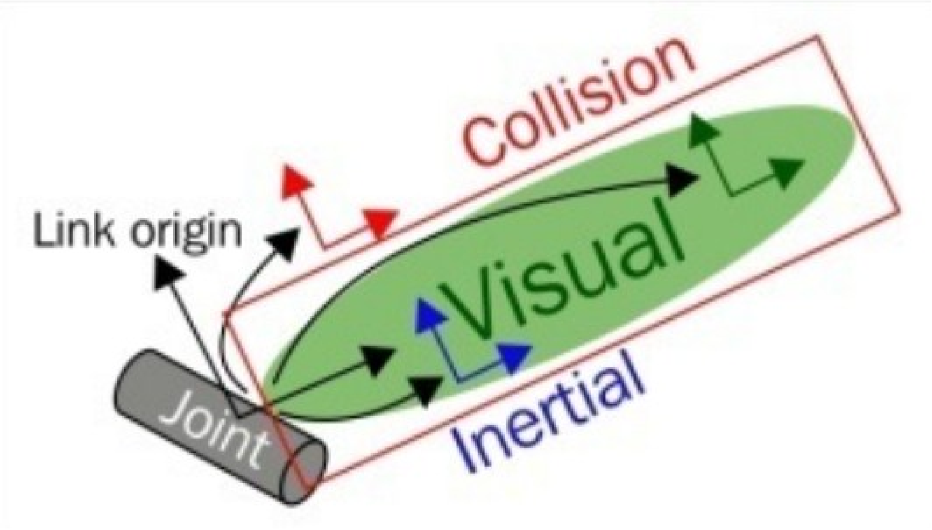
```
<link    name="name   of   the link>">
    <inertial>...........</inertial>
    <visual>    ............</visual>
    <collision>..........</collision>
</link>
```

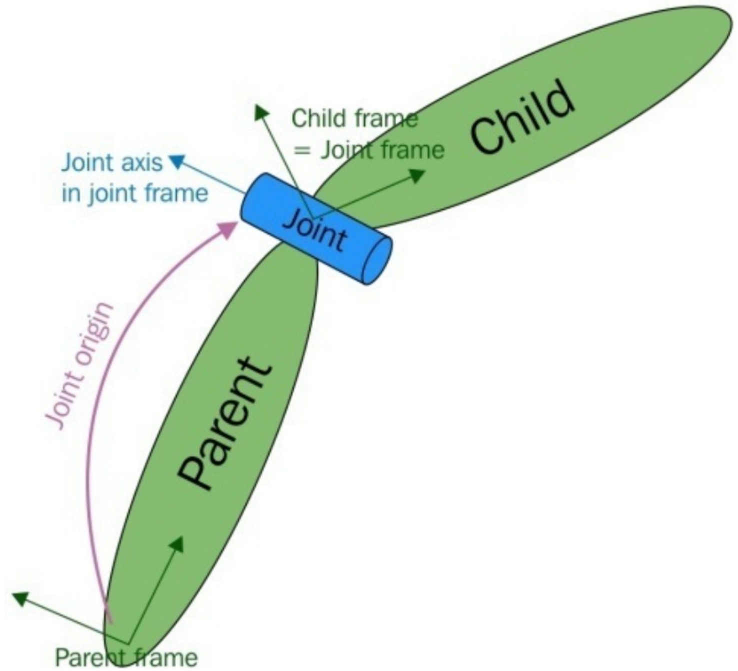


Some URDF Tags

- **Joint:**
  - Actual join of robot
  - Specify Kinematics and dynamics of joint.
  - Set joint movements and velocity
  - Joint is formed between **Parent** link and **Child** link.
  - Revolute, continuous, prismatic, fixed.

```
<joint  name="name  of  the joint>">
        <parent link="link1"/>
        <child  link="link2"/>

        <calibration />
        <dynamics />
        <limit  />
</joint>
```



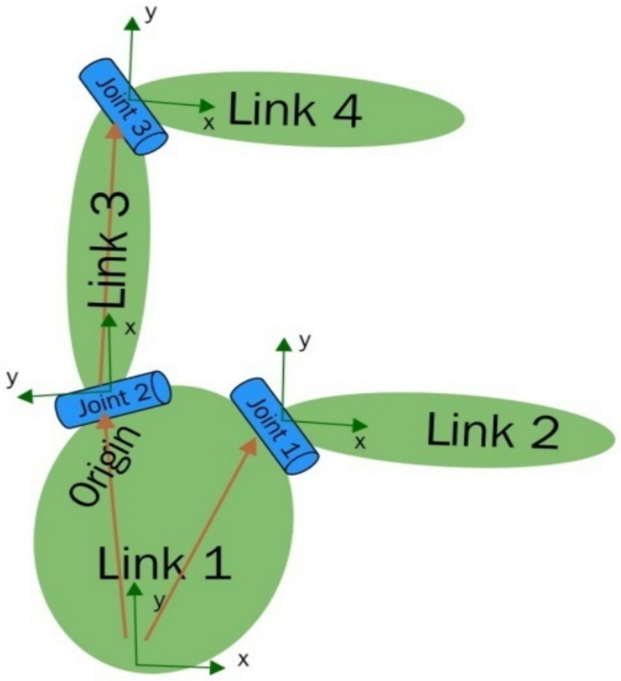


Some URDF Tags

- **Robot:**
  - Encapsulates the entire robot model
  - Name, links, joints

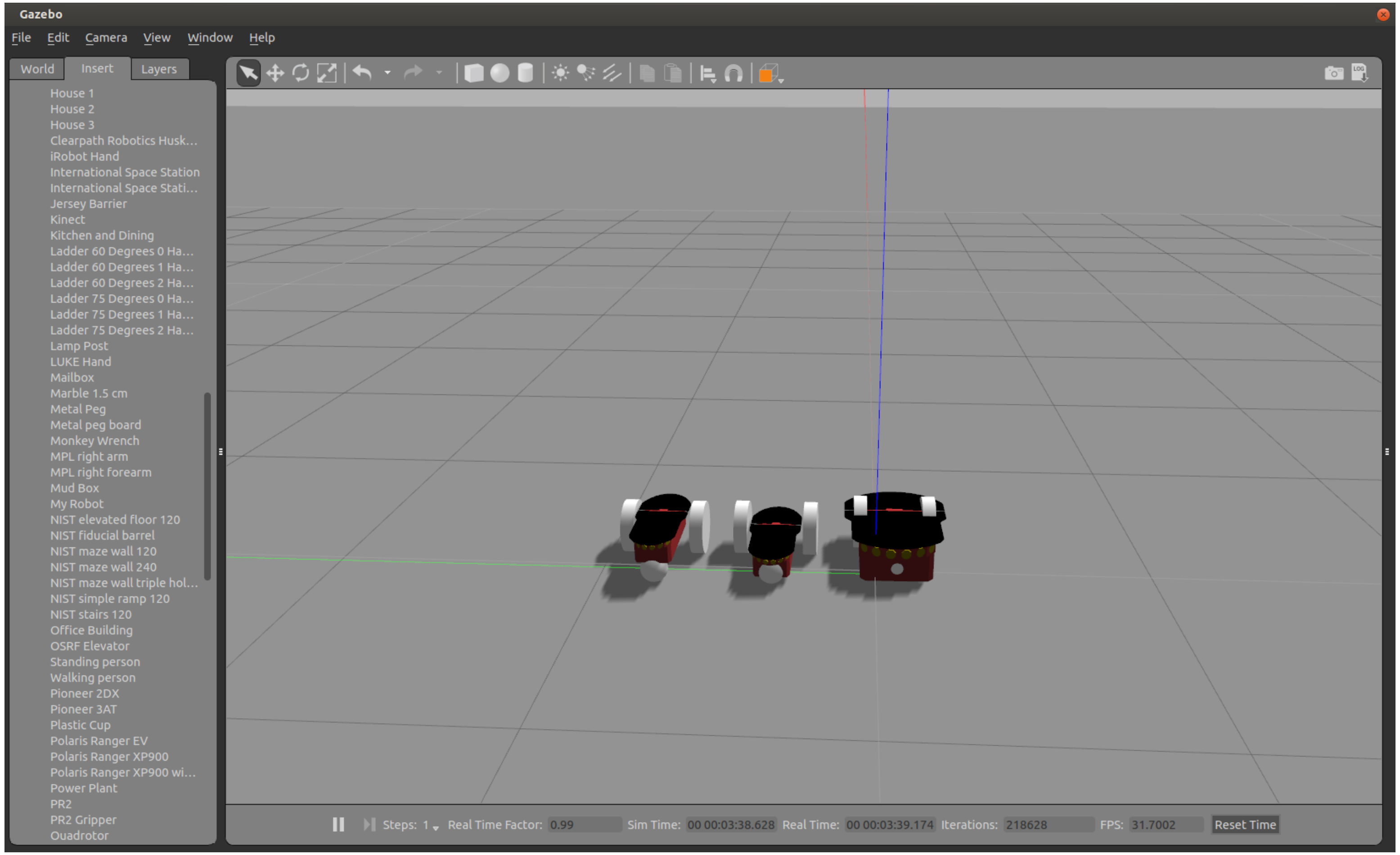
```
<robot   name="name  of  the robot>"
         <link>     .....   </link>
         <link>  ......  </link>
         <joint> ....... </joint>
         <joint> ........</joint>
</robot>
```

You can find more URDF tags at
http://wiki.ros.org/urdf/XML.



Getting Started

- Open a terminal
- type "gazebo"
- Launch any model.

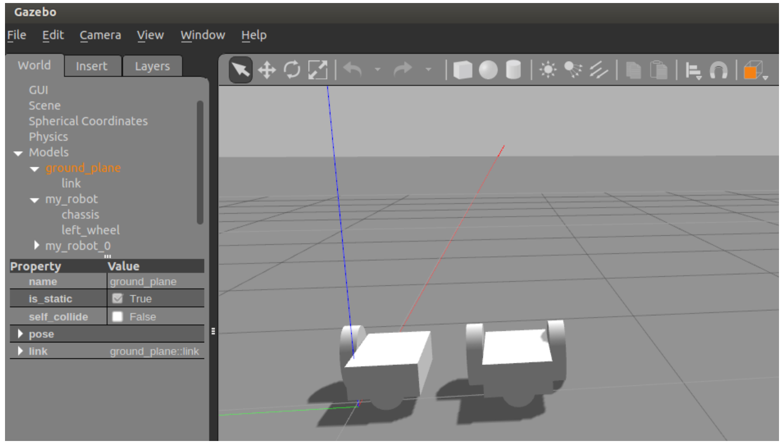


Task 1: Perform Following Gazebo Tutorials

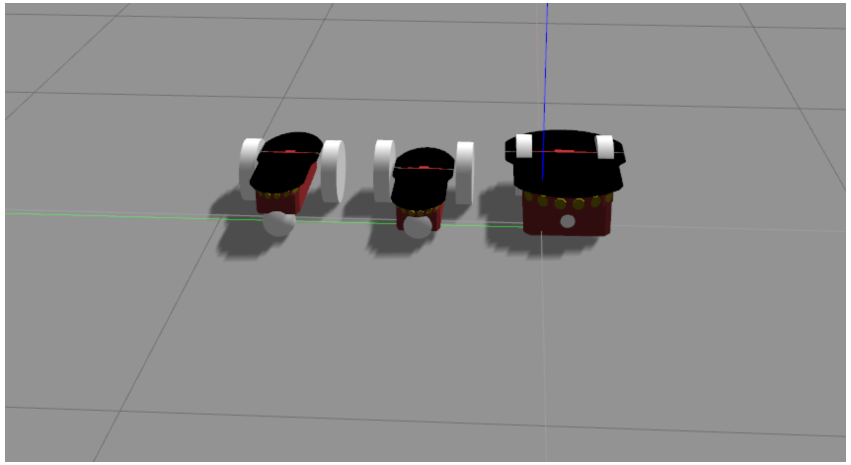
- Build a Robot
  - [Make a Mobile Robot]
  - [Import Meshes]
  - [Attach Meshes]
  - [Add a Sensor to a Robot]

Task-1: Make a Mobile Robot

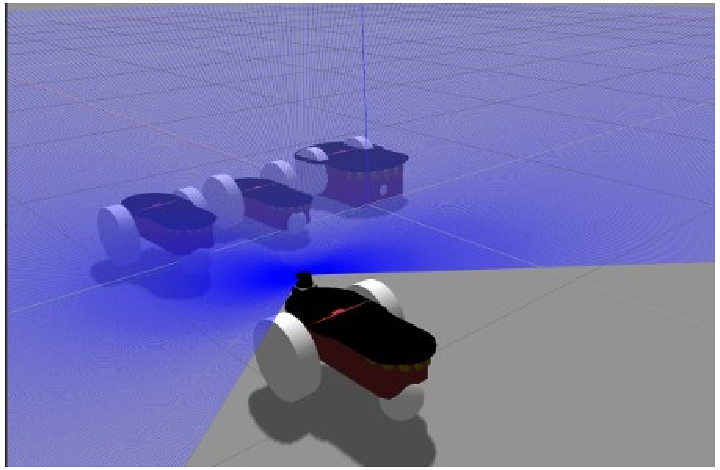
- Build the Model's Structure



Attach Meshes



- Adding a Sensor(Laser)



Task 2: Perform Following Gazebo Tutorials

- Build a World
  - Building a world
- Write a plugin
  - Plugins 101
  - Model plugins
  - World plugins
- Sensors
  - Sensor Noise Model (Ray Laser noise)

Task 3: Perform Following Gazebo Tutorials

- Connect to ROS
  - Installing Gazebo_ros_pkgs
  - using roslaunch
  - Gazebo Plugins in ROS
    - Adding Plugins
    - Differential Drive
  - ROS communication
  - ROS Plugins
- Use Rviz to visualize odometry and laser scan topics.

Lab Assignment (due before next lab)

1. Create a ROS node to communicate with robot odometry and laser range scanner data. Use the robot wheel odometry to estimate the wheels velocity (Hint: inverse kinematics). To navigate the robot use existing teleop node.
2. Build a 4-wheel Ackermann steering robot in gazebo using model files. Use existing plugins to drive the robot.
3. Bonus: create your own plugin for the robot drive system