Gazebo: Build a Robot

- Gazebo is able to dynamically load models into simulation either programmatically or through the GUI.
- Models in Gazebo define a physical entity with dynamic, kinematic, and visual properties.
  - In addition, a model may have one or more plugins, which affect the model's behavior.
- A model can represent anything from a simple shape to a complex robot; even the ground is a model.
- Gazebo relies on a database to store and maintain models available for use within simulation.
  - https://bitbucket.org/osrf/gazebo_models {Bitbucket repository}

★Model database Structure

A model database must abide by a specific directory and file structure.

The root of a model database contains one directory for each model, and a database.config file with information about the model database.

- *Database*
  - *database.config* : Meta data about the database. This is now populated automatically from CMakeLists.txt
  - *model_1* : A directory for model_1
    - *model.config* : Meta-data about model_1
    - *model.sdf* : SDF description of the model
    - *model.sdf.erb* : Ruby embedded SDF model description
    - *meshes* : A directory for all COLLADA and STL files
    - *materials* : A directory which should only contain the `textures` and `scripts` subdirectories
      - *textures* : A directory for image files (jpg, png, etc).
      - *scripts* : A directory for OGRE material scripts
    - *plugins*: A directory for plugin source and header files

database.config

This file contains license information for the models, a name for the database, and a list of all the valid models.

Note: The database.config file is only required for online repositories. A directory full of models on your local computer does not need a database.config file.

The format of this `database.config` is:

```
<?xml version='1.0'?>
<database>
  <name>name_of_this_database</name>
  <license>Creative Commons Attribution 3.0 Unported</license>
  <models>
    <uri>file://model_directory</uri>
  </models>
</database>
```

Model.config

- Each model must have a model.config file in the model's root directory that contains meta information about the model.
- The format of this `model.config` is:

```
<?xml version="1.0"?>

<model>
  <name>My Model Name</name>
  <version>1.0</version>
  <sdf version='1.5'>model.sdf</sdf>

  <author>
    <name>My name</name>
    <email>name@email.address</email>
  </author>

  <description>
    A description of the model
  </description>
</model>
```

Model.sdf

- Each model requires a model.sdf file that contains the Simulator Description Format of the model.

Model.sdf.erb

- Standard SDF file which can contain ruby code embedded.
- This option is used to programatically generate SDF files using Embedded Ruby code templates.
- Ruby conversion should be done manually (erb model.sdf.erb > model.sdf) and the final model.sdf file must be uploaded together with the model.sdf.erb (this one only for reference).

★ How to contribute a model

- Note: You don't need to add your model to the database in order to use it in Gazebo. The database is a common place where you can find models which are useful for the whole community.

Creating a model

- Create a directory for your model under the gazebo_models directory.
- That directory must include the file model.config and at least one .sdf file. It may include other files as well (meshes, textures, templates, etc.)
- Also make sure you add the model directory to the CMakeLists.txt file.

Create Pull request

Make a model

SDF Models can range from simple shapes to complex robots.

It refers to the <model> SDF tag, and is essentially a collection of links, joints, collision objects, visuals, and plugins.

Components of a SDF model

1. Link

- A link contains the physical properties of one body of the model.
- This can be a wheel, or a link in a joint chain.
- Each link may contain many collision and visual elements.
- Try to reduce the number of links in your models in order to improve performance and stability.

a. Collision

- A collision element encapsulates a geometry that is used to collision checking.
- This can be a simple shape (which is preferred), or a triangle mesh (which consumes greater resources).

b. Visual

- A visual element is used to visualize parts of a link.
- A link may contain 0 or more visual elements.

c. Inertial

- The inertial element describes the dynamic properties of the link, such as mass and rotational inertia matrix.

d. Sensor

- A sensor collects data from the world for use in plugins.
- A link may contain 0 or more sensors.

e. Light

- A light element describes a light source attached to a link.
- A link may contain 0 or more lights.

2. Joint

- A joint connects two links.
- A parent and child relationship is established along with other parameters such as axis of rotation, and joint limits.

3. Plugin

- A plugin is a shared library created by a third party to control a model.

Building your model

1. Collect your mesh

   → This step involves gathering all the necessary 3D mesh files that are required to build your model.

   → Gazebo requires that mesh files be formatted as STL, Collada or OBJ, with Collada and OBJ being the preferred formats.

2. Make your model Sdf file

3. Add to the model Sdf file

Meshes can add realism to a model both visually and for sensors.s