DRC Kickoff: Gazebo Workshop

October 25, 2012
Open Source Robotics Foundation
Table of Contents

Overview
Features and Capabilities
Installation and Execution
World and Model Creation
  Break (10:20 - 10:40am)
Plugin Development
Cloud Simulation
  Lunch Break (12:00 - 1:30pm)
Tutorials
  Exercise 1: Build a Model
  Exercise 2: Control a Model
  Exercise 3: Build a World
    Break (3:00pm - 3:20pm)
  Exercise 4: ROS Integration
  Exercise 5: DRC Simulator
Q&A (4:45pm - 5:00pm)
Overview
Overview

Objectives
• Describe Gazebo
• Introduce the DRC Simulator
• Provide hands-on experience

Audience Participation
• This is not a lecture, please ask questions

Support Material
• Workshop material will be online
  http://gazebosim.org
The Gazebo Simulator

History
• Created at USC as part of the Player project
• Originally designed for simulation of outdoor mobile robots
• Relies on external robot control software: ROS, Player

Purpose
• Simulate the real world
• Test and develop hardware and software
• Regression testing
• Research
  • Human-Robot Interaction
  • Advanced physics simulation
  • Common test suite
Use Cases
Virtual Robotics Challenge (VRC)

Host a simulated version of the DRC
- Common infrastructure
  - Cloud-based simulation

Provide simulated task environments
- Drive a vehicle
- Walk across simple and difficult terrain
- Attach a fire hose to a stand pipe

Post-DRC
- Continue development of Gazebo
Gazebo, ROS, and DRC Simulator

Gazebo
• Current version: 1.2.5
• Stand-alone application

ROS
• Current version: Fuerte, Groovy soon-to-be-released
• Uses Gazebo 1.0

DRC Simulator
• Current version: 1.0.2
• Uses and builds on the stand-alone version of Gazebo
• Uses ROS Fuerte to import and control DRC robot model

Your Code
• May use any robot control software: ROS, Player, custom
Roadmap

DRC Sim 2.0 (Gazebo 1.3) - February 2013
• Real-time performance of DRC robot
• Data logging and playback
• Mechanical vehicles control
• Multiple floor buildings, rubble piles
• Bullet integration

DRC Sim 2.5 (Gazebo 1.4) - April 2013
• GUI model creation
• Integrate validation results
• Mechanical tool interface

DRC Sim 3.0 (Gazebo 1.5) - August 2013
• Digital Elevation Models (DEM)
• Shared memory interface
Gazebo Features
Physics

Multiple Physics Engines
- Abstract interface layer between Gazebo and physics engines
- Near-term: ODE and Bullet
- Far-term: Simbody, DART, Moby, and others

Selective dynamics control
- Physically simulate only parts of a model
- Reduce computation
- Tip: modeling behavior, rather than a kinematically correct robot, is often sufficient

Future
- Friction models
- Noise models
Rendering

Sensor visualization
- Projected camera views
- Laser rays
- RFID range as translucent spheres

Custom GUI overlays
- Create unique interfaces using CEGUI

Sky
- Sun, moon, stars
- Volumetric clouds

Future
- User-defined visualizations
- Improved fidelity
Transport

Message passing
- Google Protobuf
- boost::asio provide socket based comms

Server and client separation
- Many clients, one server
- Custom clients

Future
- Shared memory interface
Model Database

Central repository for all models

- **Location:** [http://gazebosim.org/models](http://gazebosim.org/models)
- Meshes, textures, plugins
- Accessible through GUI
  - Drag-and-drop models into a running simulation
- Format fully documented
  - Create your own model database

Future

- Web-based tool for browsing and contributing models
- World database
Graphical User Interface

Control joints
- Apply force to joints
- Position and velocity PID controllers

Manipulate pose
- Drag models, snap to grid

Modify running simulation
- World and model parameters

Future
- Position end effectors using the mouse
- Apply forces and torques to models
- Build models
- Custom visualizations
Documentation

API
• http://gazebosim.org/api
• Doxygen generated

Simulation Description Format (SDF)
• http://gazebosim.org/sdf
• Fully featured and scalable XML description format for robots and simulation

User guide
• http://gazebosim.org/user_guide
• A written guide to installing and using Gazebo

Tutorials
• http://gazebosim.org/wiki/tutorials
• Building models, worlds, plugins
• Using the DRC simulator
Reliability

Static code checking
• cppcheck
• cpplint

Dynamic code checking
• gcc uses most compile time warnings

Regression testing
• Covers physics, math library, sensors, transport layer

Continuous integration
• Jenkins: http://build.osrfoundation.org

Validation
• Working with NIST to validate DRC robot and environments
Requirements

Hardware
• Modern GPU: nVIDIA preferred, less than 4 years old
• Multi-core CPU

Required software
• OGRE - rendering engine
• Boost - threading, transport, command-line parsing
• Protobuf - message serialization
• TinyXML - XML parser
• libCurl, libtar - model database extraction
• FreeImage - heightmap

Optional software (Installed with DRC Simulator package)
• URDF DOM - import URDF files
Installation Methods

Options

- Gazebo: stand-alone version
- DRC Sim: Gazebo 1.2
- ROS: Use ROS installation method
  - Fuerte: Gazebo 1.0.2
  - Groovy: Gazebo 1.0.2

Install

- Ubuntu 12.04 - APT repository
- Other Linux distributions - Tarball available on http://gazebosim.org
- Developers - Mercurial repository hosted on Bitbucket

Future installation methods

- RPM-based Linux distributions
- OSX
- Windows
Installation Methods
DRC Simulator Dependency Graph

ROS Fuerte

- pr2_simulator
- simulator_gazebo

Gazebo 1.0.x
- ros_comm
- urdfdom
- visualization_common

OGRE 1.7.3 w/ Cg support

drcsim 1.0.x
- gazebo 1.2.x
Getting / Giving Help

Questions and answers (http://answers.gazebosim.org)
• Search, post or answer questions about Gazebo

Wiki (http://gazebosim.org/wiki)
• Tutorials and general documentation

API (http://gazebosim.org/api)
• Doxygen code documentation

Mailing list (http://kforge.ros.org/mailman/listinfo/gazebo-list)
• Announcements

Bug tracker (http://bitbucket.org/osrf/[gazebo | drcsim]/issues)
• Find and issue bugs, and request new features

Source Code (http://bitbucket.org/osrf/[gazebo | drcsim])
• For developers, Mercurial source checkout and install
Contributing to Gazebo

Step 1: Communicate
• Use mailing list to find out who is working on what
• Announce new models, and features

Step 2: Develop
• Fork Gazebo from Bitbucket

Step 3: Review
• Create a pull request
• Gazebo team will review code, and offer feedback

Step 4: Merge
• Once pull request is accepted
Creating Worlds
Elements within Simulation

World
• Collection of models, lights, plugins and global properties

Models
• Collection of links, joints, sensors, and plugins

Links
• Collection of collision and visual objects

Collision Objects
• Geometry that defines a colliding surface

Visual Objects
• Geometry that defines visual representation

Joints
• Constraints between links

Sensors
• Collect, process, and output data

Plugins
• Code attached to a World, Model, Sensor, or the simulator itself
Element Hierarchy

- World
  - Scene
  - Physics
  - Model
    - Link
    - Collision
    - Visual
    - Sensor
      - Plugin
        - Plugin
          - Plugin
          - Light
Element Types

Collision and Visual Geometries
- Simple shapes: sphere, cylinder, box, plane
- Complex shapes: heightmaps, meshes
Joint Types

Prismatic: 1 DOF translational

Revolute: 1 DOF rotational

Revolute 2: Two revolute joints in series

Ball: 3 DOF rotational

Universal: 2 DOF rotational

Screw: 1 DOF translation, 1 DOF rotational
Sensors and Lights

Sensors

- Ray: produces range data
- Camera (2D and 3D): produces image and/or depth data
- Contact: produces collision data
- RFID: detects RFID tags
  
User contributed

Lights

- Point: omni-directional light source, a light bulb
- Spot: directional cone light, a spot light
- Directional: parallel directional light, sun
Types of Worlds

**Simple**
- Focused scenario
- Manipulation
- Perception

**Indoor**
- Path planning
- Mobile manipulation
- Clone real environment

**Outdoor**
- Aerial robots
- Outdoor mobile and legged robots
System Components

Physics Library
• Loads and runs the dynamics engine

Sensor Library
• Generates sensor data

Rendering Library
• Draws the world for the GUI and Sensor Library

Transport and Messages Library
• Implements socket-based connections for message passing

Math and Common Libraries
• Internal math functions, and shared utilities

GUI and Command line tools
• Executables to visualize and manipulate simulation
World Definition

Simulation Description Format (SDF)
- http://gazebosim.org/sdf
- XML-based format that describes models and environments

Graphical Interface
- Import models (model database)
- Place models
- Manipulate models
- Save worlds

Simulation Units
- Controllable simulation speed
- Choosing consistent set of simulation units: MKS
World Creation Demo
What is a Model

Any collection of links, joints, sensors and plugins
Robot Models

Simple platforms
- Built-in shapes
- Mesh skinning

Realistic physical properties
- Meshes as collision objects
- Mass and inertia properties
- Surface friction
- 6 joint types

Full sensor suite
- Laser range finders
- Mono/Stereo cameras
- Kinect
- Contact
- Joint force/torques
Non-Robot Models
Building a Model

**Step 1: Collect meshes**
- Make custom meshes: Sketchup, Blender
- Export from Solid Works (http://ros.org/wiki/sw_urdf_exporter)
- Online repositories: 3D Warehouse

**Step 2: Make an SDF file**
- Start simple
- Add links and collision elements one at a time
- Add joints last

**Step 3: Include Model in a World**

```
<include>
  <uri>model://my_model</uri>
</include>
```

**Step 4: Share your model**
- Add to model database
Step 1: Collect Meshes

Reduce complexity
• Meshes with low polygon count are more efficient
• Use normal maps for improved visualization

Center the mesh
• Move the center of the mesh to (0, 0, 0) in editor before export to Collada or STL
• This will simplify placement within Gazebo

Scale the mesh
• Make sure the mesh is in meters, and sized properly
Step 2: Make an SDF File

Step 2a: Static model
• Skips physics update, and allows easy placement of model components

Step 2b: Add each link
• Add collision and visual objects for each link
• Test your model with each addition

Step 2c: Add each joint
• Remove static constraint
• Reduce joint count, and test each joint

Step 2d: Add sensors
• Connect sensors to appropriate links

Step 2e: Add plugins
• Add plugins to control joints and sensors

Step 2f: Test and tune
Efficient Models

Static models
• Not dynamically-simulated
• Act only as collision objects
• Static models can be animated

Reduce joints
• Add collision and visual objects for each link
• Test your model with each addition
Contributing Models

Model Repository
• Hosted on Bitbucket
• https://bitbucket.org/osrf/gazebo_models
• Fork the repository, add your model, submit a pull request

Create a new repository
• Follow the model database format
• Advertise your repository on the Gazebo mailing list

Mailing list
• See who is creating models
• Ask for help
Model Creation Demo
Plugin Development
Plugin Overview

Direct access to Gazebo API
• Location: http://gazebosim.org/api

Easily Shared
• Model database

Dynamically-Loaded
• Insert and remove from a running system

Examples
• Tutorials: http://gazebosim.org/Tutorials
• Source: <gazebo_source>/plugins
Plugin Types

World
• Plugin receives a pointer to the World instance

Model
• Plugin receives a pointer to a Model instance

Sensor
• Plugin receives a pointer to a Sensor instance

System
• Plugin receives the command line arguments
Plugin Types

Plugin Basics

• Inherits from appropriate parent plugin class.

```cpp
namespace gazebo
{
    class MyPlugin : public WorldPlugin { }
}
```

• Parent objects and parameters are provided via Load function.

```cpp
namespace gazebo
{
    class MyPlugin : public WorldPlugin
    {
        MyPlugin::Load(physics::WorldPtr _world,
                        sdf::ElementPtr _sdf) { }
    }
    GZ_REGISTER_WORLD_PLUGIN(MyPlugin)
}
```
World Plugin

Usage
• Include in SDF file
  
  `<world name="default">
    <plugin name="my_plugin" filename="libmy_plugin.so"/>
  </world>`

Finding Plugins
• `GAZEBO_PLUGIN_PATH` environment variable tells Gazebo where to look for plugins

Purpose
• Access to all models
• Control physics engine
World Plugin Demo
Model Plugin

Usage
• Include in SDF file
  <model name="my_model">
    <plugin name="my_plugin" filename="libmy_plugin.so"/>
  </model>

Purpose
• Control model behavior
  Joints, sensors, link pose
Model Plugin Demo
Sensor Plugin

Usage
• Include in SDF file
  
  <sensor name="my_sensor">
    <plugin name="my_plugin" filename="libmy_plugin.so"/>
  </sensor>

Purpose
• Gather and modify sensor data
  • Add noise models

Examples
• ROS laser plugin
  • ROS camera plugin
System Plugin

Usage
• Command line only
  
gzserver -s <plugin_library_file>
gzclient -g <plugin_library_file>
gazebo -s <plugin_library_file> -g <plugin_library_file>

Purpose
• Modify system paths (resources, plugins, models)
• Control Gazebo bring-up, execution and shutdown

Examples
• ROS API system plugin
Distributed Simulation Environment
Distributed Simulation Environment

Purpose
• Simulation in the cloud
• Separate client from server
• Run on separate machines

How it works
• Master
  Tracks server and clients
• Server
  Run physics simulation
  Generate simulated sensor data
• Client(s)
  Visualize worlds
  User interactions
Running the Master, Server, Client

Specify Master
• Environment variable
  GAZEBO_MASTER_URI=http://localhost:11345

Server
• Automatically starts a master if none present
  Command: gzserver <world_filename>

Client
• World visualization is the most common use case
  Command: gzclient
Running on a Local Network

Step 1
• Set GAZEBO_MASTER_URI=http://localhost:11345 on all machines
  export GAZEBO_MASTER_URI=http://localhost:11345

Step 2
• Start a server on one machine
  gzserver <world_filename>

Step 3
• Start a client on a different machine
  gzclient
Debugging

Command line tools

• **gzstats**
  A client that prints out info from a server
  Check general server performance status

• **gztopic**
  A client that lists all topics that are active
  Debug message passing between server and client
Cloud Simulation

Web interface to cloud machines
- Automatically launch Gazebo instances
- Teams may request configuration and time

Scale-up access over a few months
Lunch Break

1 hour

After lunch: Q&A, Tutorials
Post Lunch Question and Answer
Exercise 1: Building a mobile robot
Exercise 1
Overview

Topics Covered
• Construction of a two-wheeled mobile base
• Attaching meshes to visual elements
• Attaching sensors to links
• Constructing a simple gripper
• Attaching a gripper to a mobile base

Wiki Tutorials
http://gazebosim.org/wiki/Tutorials
Section: Building a Robot
Exercise 2: Controlling a mobile robot
Exercise 2
Simulation Controls Overview

Animation vs. Dynamic control

• Animation
  Fast.
  Disregard physics, constraints*.
  No collision responses.

• Dynamic control
  Velocity control - leveraging integrator only
  Force control - leveraging physics engine ($f = ma$)
  Can be computationally intensive

• Controllers with sensor feedback.
• Gazebo's built-in PID class.
Exercise 2
Simulation Controls Overview

Topics Covered
• Animating pose of rigid body links with the animation engine.
• Controlling pose of rigid body links by setting velocities.
• Controlling joints by applying forces.
• Controlling a robot with its simulated onboard sensor.
• Controlling a joint with Gazebo's builtin PID class.

Wiki Tutorials
http://gazebosim.org/wiki/Tutorials
Section: Controlling a Robot
Exercise 3: Building a world
Exercise 3
Overview

Topics Covered
• Constructing a world using the graphical interface
• Modifying world parameters
• Controlling the world via a plugin

Wiki Tutorials
http://gazebosim.org/wiki/Tutorials
Section: Making a World
Exercise 4: ROS Integration
Exercise 4
ROS Integration Overview

Gazebo in ROS or ROS in Gazebo?
• ROS wrapped thirdparty Gazebo installation (http://ros.org/wiki/simulator_gazebo)
  Fuerte ← Gazebo 1.0.x
• Gazebo standalone installation (http://gazebosim.org)

Model Description Formats: COLLADA, URDF, SDF, SRDF, YADF?
• Solidworks to URDF exporter
  http://ros.org/wiki/sw_urdf_exporter
• URDF Dependencies
  http://ros.org/wiki/urdf
  URDF support built at compile time in Gazebo 1.2.x*
    sudo apt-get install ros-fuerte-urdfdom
Exercise 4
ROS Integration Overview
Exercise 4
ROS Integration Overview

Gazebo Plugins with ROS dependencies

• For simulating ROS drivers for real robots
  http://ros.org/wiki/wge100_camera_firmware
  http://ros.org/wiki/microstrain_3dmgx2_imu
  http://ros.org/wiki/prosilica
  ...

• Using high level ROS applications with Gazebo
  http://ros.org/wiki/navigation
  http://ros.org/wiki/pr2_interactive_manipulation
  http://moveit.ros.org
  ...
Exercise 4
ROS Integration Overview

Topics Covered
• Managing ROS dependencies
• Building a Gazebo plugin with ROS

Wiki Tutorials
http://gazebosim.org/wiki/Tutorials
Section: ROS Integration
Exercise 5
DRC Robot Overview

DRC Robot Dynamics Model
• Initial URDF generated from simplified CAD model subject to change.

DRC Robot Sensor Suite
• Real sensor suite hardware TBD.
• For now, "Best guess" sensor suite.
  Hokuyo laser
  Stereo camera
Exercise 5
DRC Robot Overview

DRC Robot Dynamics Controls API

• Initial simulation tutorials "place holder" controllers derived from PR2 controllers

• Walking controllers interface TBD.
Exercise 5
DRC Robot Overview

Topics Covered

• Visualize and log sensor data with rviz and rxbag.
• DRC Robot basic joint control using PR2 mechanism controllers.
• Teleporting the DRC Robot.
• Customizing the DRC Robot world contents.
• Animating the DRC Robot with ROS JointTrajectory messages.
  (http://gazebosim.org/wiki/trajectory_msgs)

Wiki Tutorials

http://gazebosim.org/wiki/Tutorials
Section: DRC Tutorials