GAZEBO

John Hsu
Nate Koenig

ROSCon 2012

Outline

What is Gazebo, and why should you use it

Overview and architecture

Environment modeling

Robot modeling

Interfaces

Getting Help

Simulation for Robots

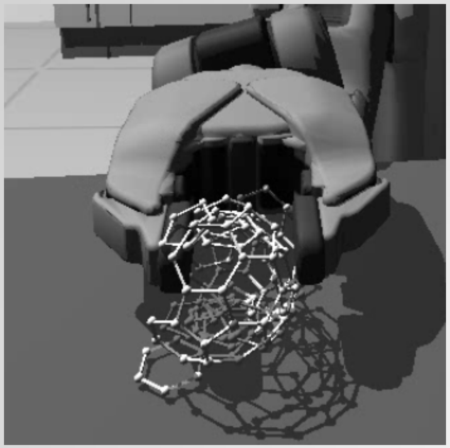
Towards accurate physical simulation

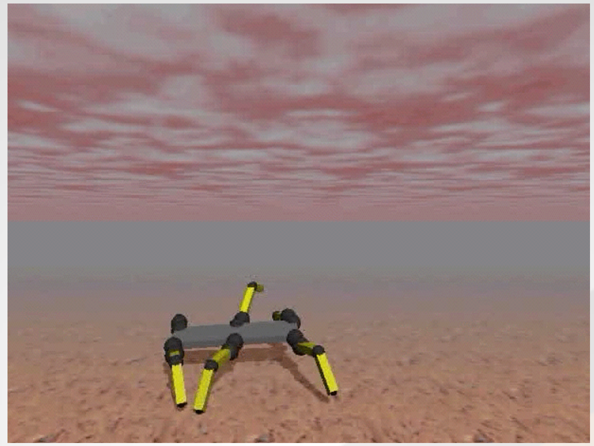
- Easy transition to and from simulation
- Remove hardware issues and resource constraints

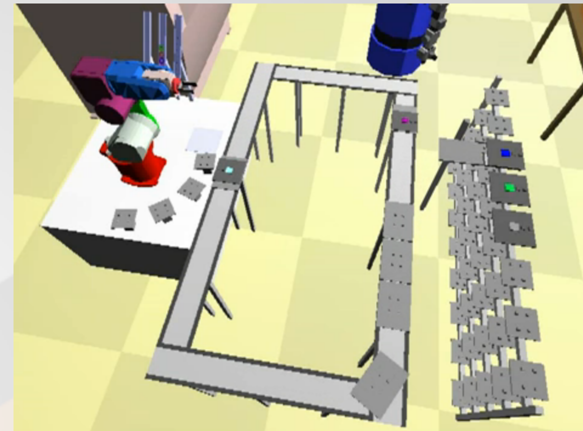
Support common robot control software

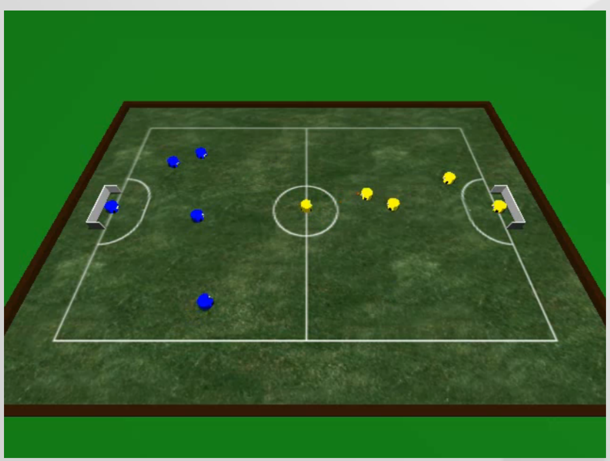
- Custom client code
- ROS interface
- Player interfaces

Use Cases









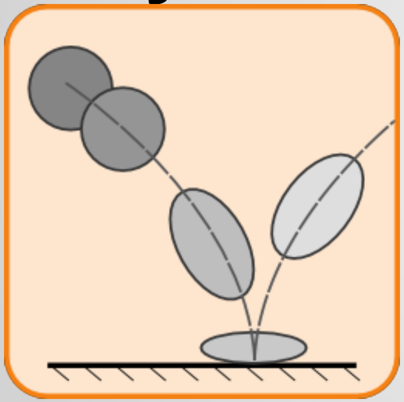
Overview & Architecture

New in 1.0

- Separation of physics and visualization
  - server: physics and sensor generation
  - client: visualization and user interface
- Socket communication
  - Protobuf provides message passing
- Simplified plugin interface
  - Control any aspect of simulation
- Simulation Description Format (SDF)
  - XML based format for worlds and models

Architecture

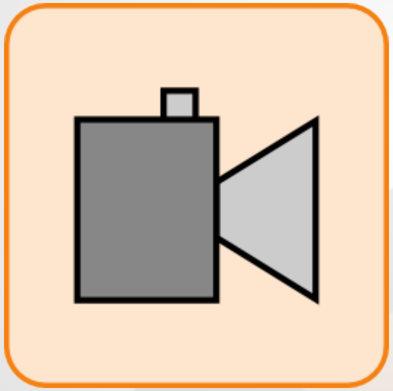
Physics



Rigid Body Dynamics

ODE
Bullet*

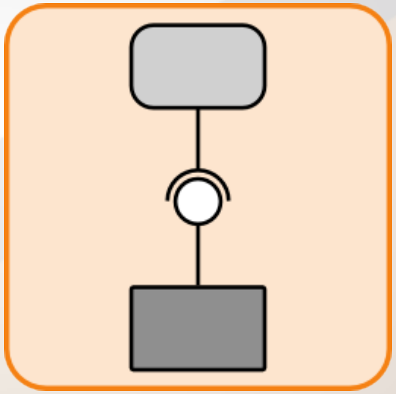
Rendering



OpenGL

OGRE

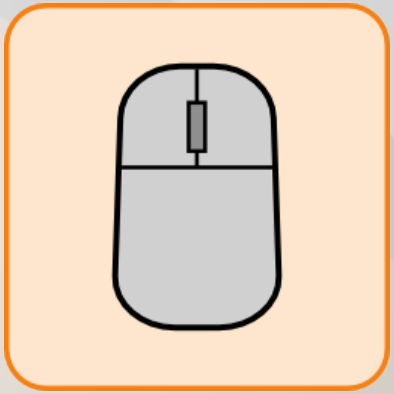
Interfaces



Plugins and IPC

Google Protobuf
Boost ASIO

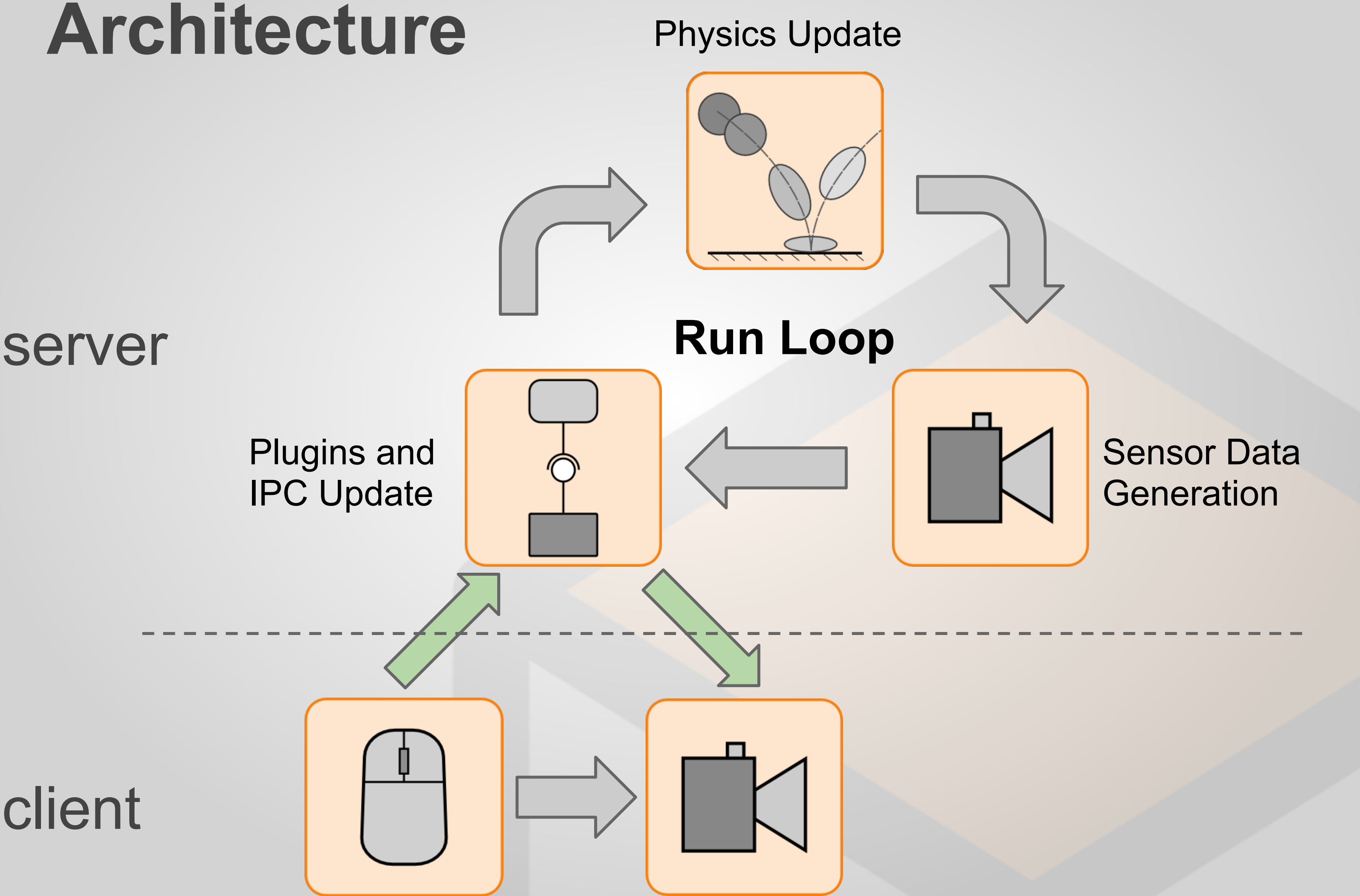
User Interfaces



GUI

QT
CEGUI

Architecture

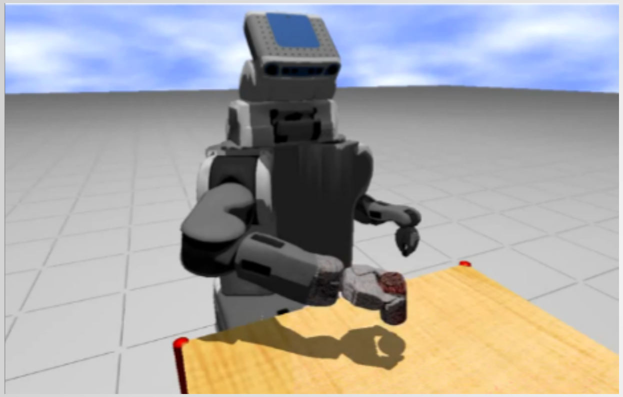


Environment Modeling

Environments

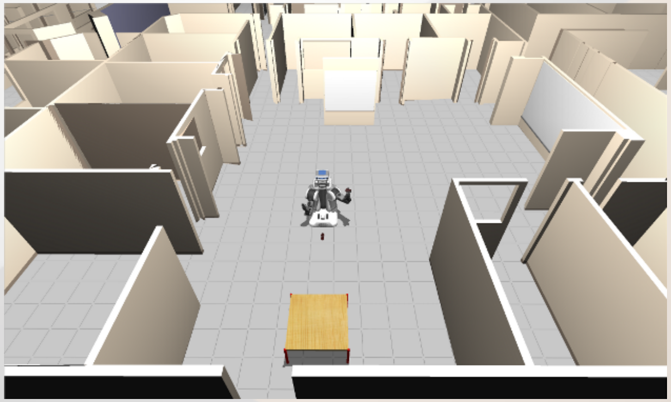
Simple

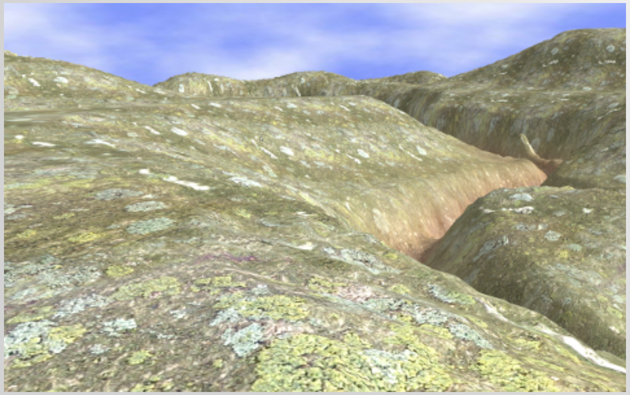
Focused scenario
Manipulation
Perception



Indoor

Path planning
Mobile manipulation
Clone real environment

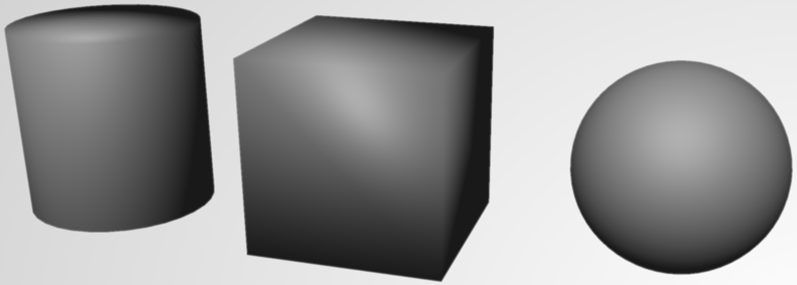




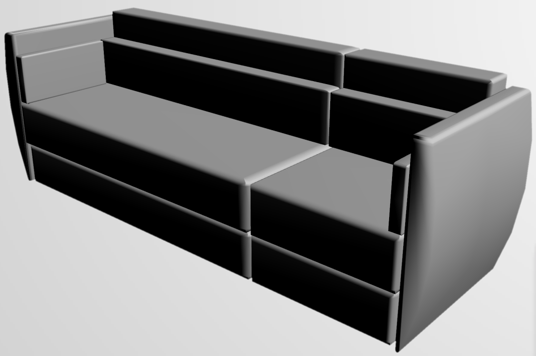
Aerial robots
Outdoor mobile and legged robots

Outdoor

Creating Environments



Built into Gazebo



3D Warehouse or model editor

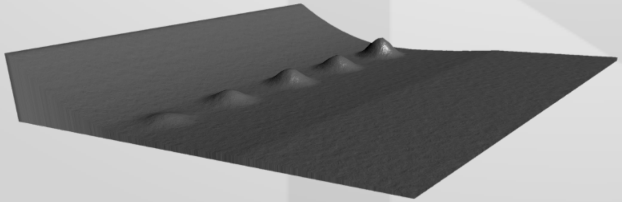


Image editor

A Word on Meshes

- Alignment and size
  - Move meshes to origin (0,0,0) when exporting
  - Stay consistent with units (preferably metric)
- Materials and lighting
  - Ambient and diffuse color properties are important
  - Lighting requires outward facing normals
  - Improve texture quality before mesh quality
- Efficient meshes
  - Reduce polygon count
  - Use normal maps for improved lighting

Organizing Resources

Directory structure for a project

Meshes: [project_path]/Media/models

Images: [project_path/]Media/materials/textures

Materials: [project_path]/Media/materials/scripts

Environment variable

export GAZEBO_RESOURCE_PATH=[project_path]

API

gazebo::SystemPaths::AddGazeboPaths(string);

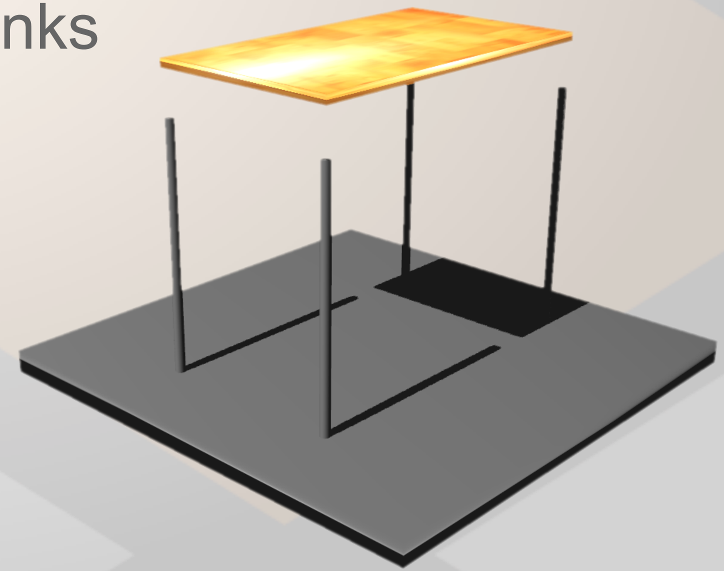
Efficient Environments

Static Models

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

Reduce Joints

- Create models using composite links



Add Visual Realism

Lighting

- Limit number of lights, and reduce ambient light
- Use directional lights for shadows
- Desired effects requires parameter tuning

Custom shaders

- Create and load vertex and pixel shader via material scripts

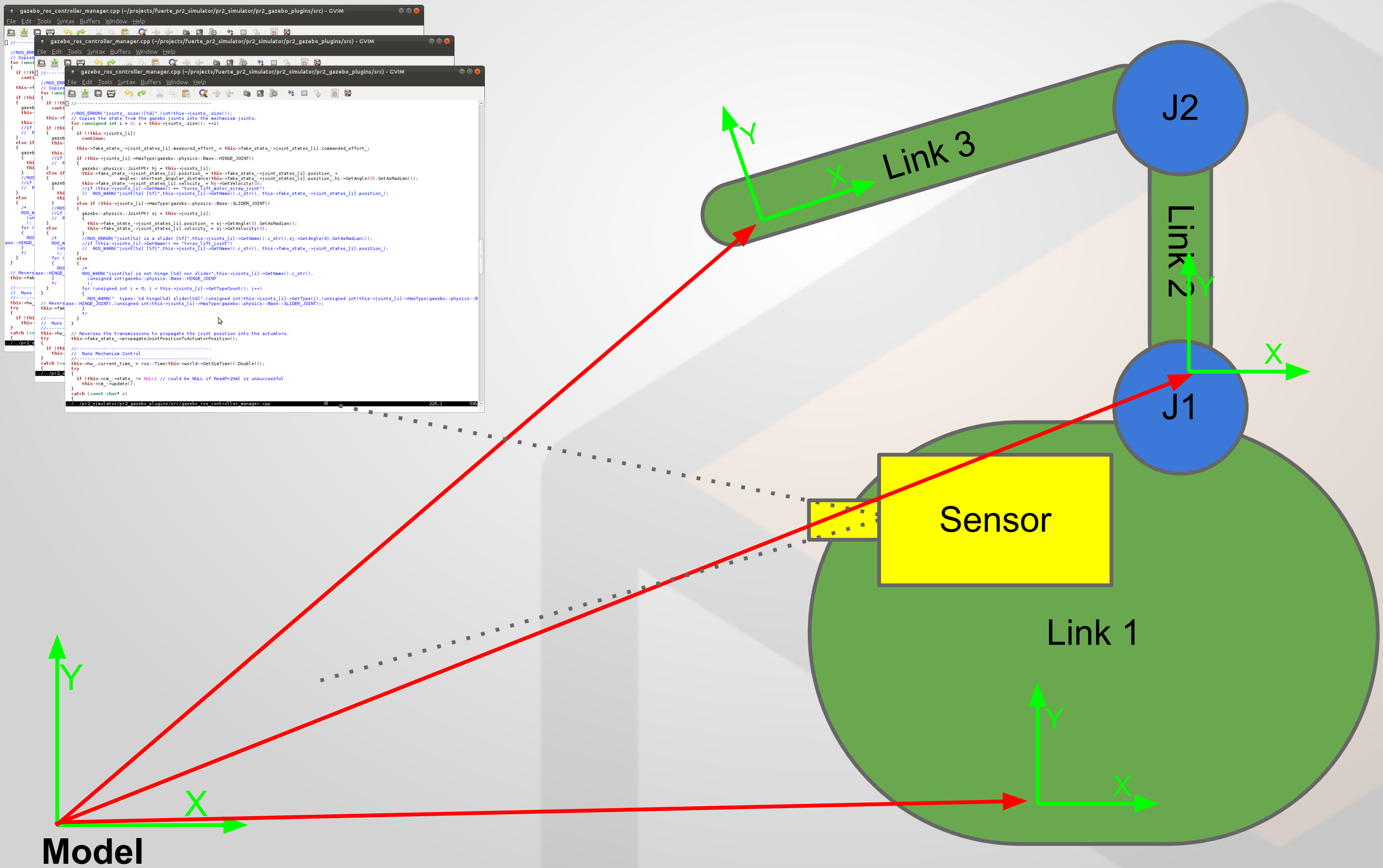
Sky and fog

- Add any material to a sky dome
- Fog can add a horizon and add sense of distance

Robot Modeling

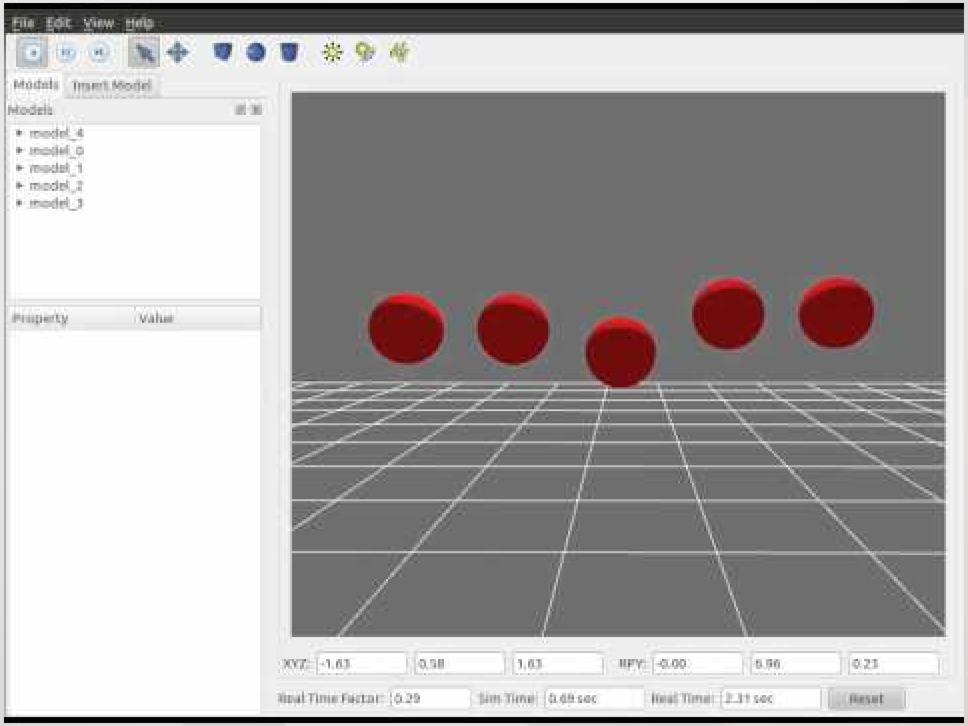
What is a Robot (Model)?

A collection of links, joints, sensors, actuators and plugins.



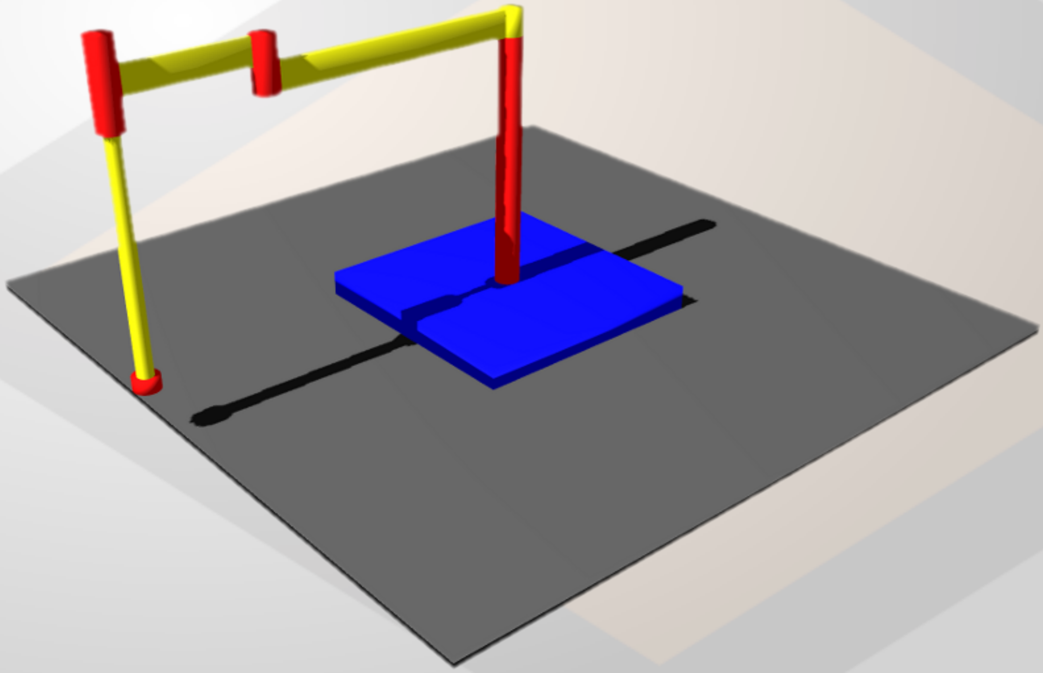
Example: Mass Spring System

Simple mass spring system in Gazebo:

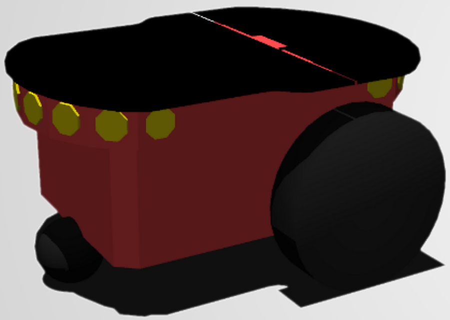


Example: SCARA Arm

Simplified arm model



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

Why 3D Dynamics Simulator

Dynamics simulation

- "Looks right" interactive mechanical behaviors
- Non-interactive higher fidelity dynamics

Visual simulation

- 3D image, range, depth sensor generation

Closing the loop between visual and dynamics simulation.

What to Expect (Dynamics)

Motion
- Newton-Euler equations.
- First order time integrator.

Constraints
- Frictionless joints.

Collision
- Perfectly inelastic collision*.

Contact
- Friction pyramid.

Modeling: URDF and SDF

How to specify a robot model

URDF format and SDF format

URDF vs SDF

URDF	SDF
• Tree • Link --> Link transforms • Link and Joint + "Extensions"	• Graph • Model --> Link transforms • Link, Joint, Sensors, Plugins, Lights, Physics, Scene.

URDF --> SDF converters

rosrun urdf2model -f <urdf> -o <sdf>

Contributing robot models

Soon to be released online model database

What are Links

Inertial (mass, moment of inertia)

The "M" in f=Ma for physics engines

Collision (geometry)

Used by collision engine to generate contact joints for the physics engine

Visual (geometry)

Used by render engine to generate images for GUI and camera or depth sensors

Joints

User defined joints

Type	DOF
revolute prismatic revolute2	1 rotational 1 translational 2 rotational

Type	DOF
universal ball screw	2 rotational 3 rotational 1 trans. 1 rot.

Dynamically created

- Contact joints between objects
  - Created from colliding collision geometries
  - Limited to 20 contacts for each colliding pair by default
  - Contact information accessible through Contact Sensor

Sensors

- Camera
  - Render to offscreen buffer
- Kinect
  - Depth camera
- Laser
  - CPU and GPU based ray casting
- Contact
  - Generated by collision engine
- RFID
  - Information generated from model positions
- Force torque
  - Specific to joints at the moment

Efficient Robot Models

Physics (CPU):

- Limit contacts (<physics max_contacts="3"/>)
- Kinematic trees are better than loops
- Reduce number of joints in a model

Collision (CPU):

- Primitives are more efficient than trimeshes
- Limit collision mesh size (< ~5k triangles per link)

Rendering (GPU):

- Limit visual mesh size (< ~5k triangles per link)
- Limit image/depth sensor resolution or rate

How to Improve Dynamics Accuracy

...with maximal (Cartesian) coordinate solvers such as ODE or Bullet.

How to choose time step size:

- Motor controller frequency driven.
- First order Euler time stepping $O(\Delta t)$.

How to tweak solver parameters:

- <solver type="quick" **iters="100"**/>
- Default 10 iterations for LCP solve, increase if necessary.

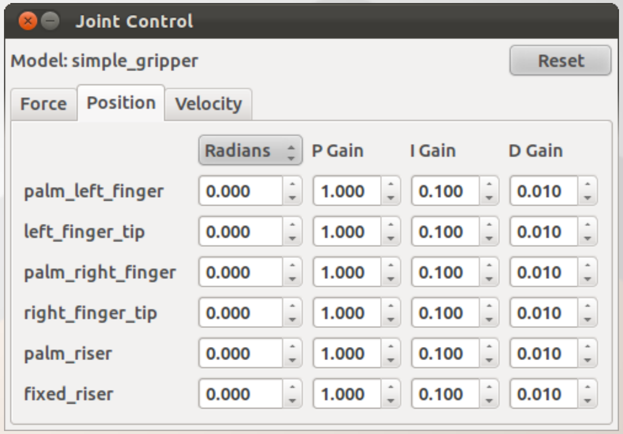
Model physics of the real robot more closely

- Account for more details. E.g. prismatic vs. screw.

Controlling the Robot Model

Graphical joint control widget in Gazebo

- Direct force control.
- PID position and velocity.



Programmatic control

- Level of abstraction, hardware/software transparency.
- World plugins: access to all models.
- Model plugins: access to all joints and links.

Troubleshooting Models

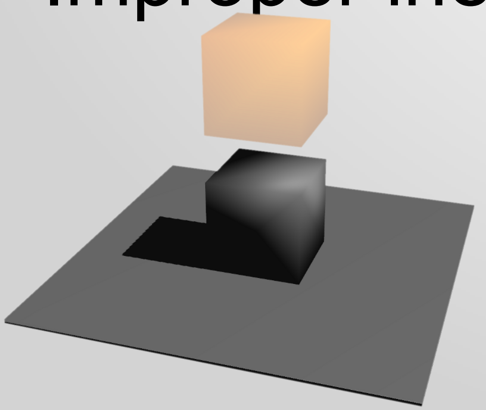
Mesh is out of place or has improper scale

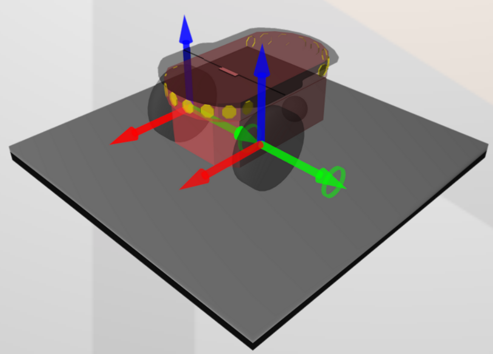
- Recenter and scale mesh using 3D modeling application
- Enable "Show Collisions" in GUI to debug

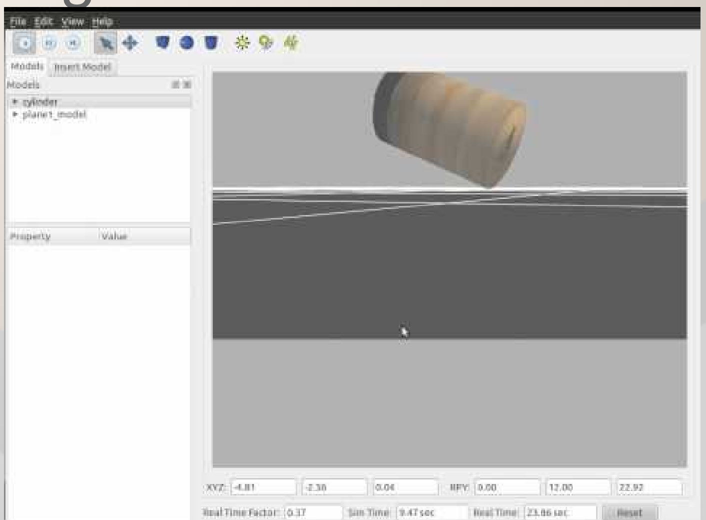
Improper joint placement and rotation

- Enable "Show Joints" in GUI to debug

Improper inertial values





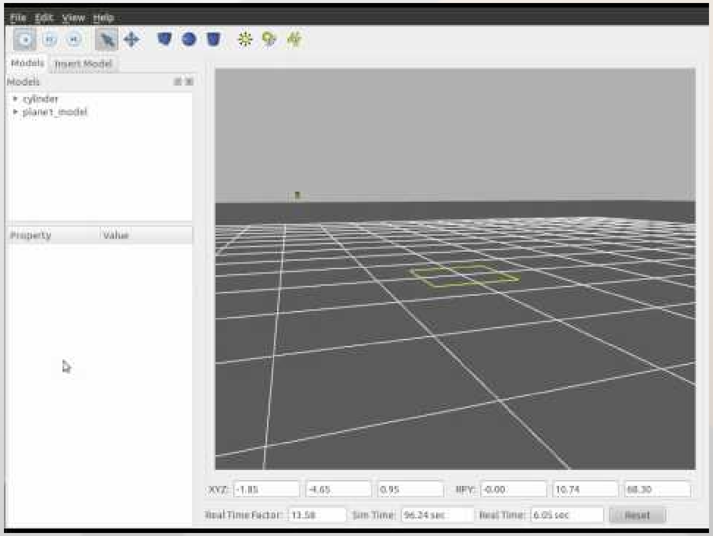


Troubleshooting Models

Symptom: Model flies away, spins out of control

Cause: Interpenetration with surroundings

Solution: Step through simulation slowly. Check for collisions, interpenetrations between model/ground. Spawn model away from other objects.



Troubleshooting Models

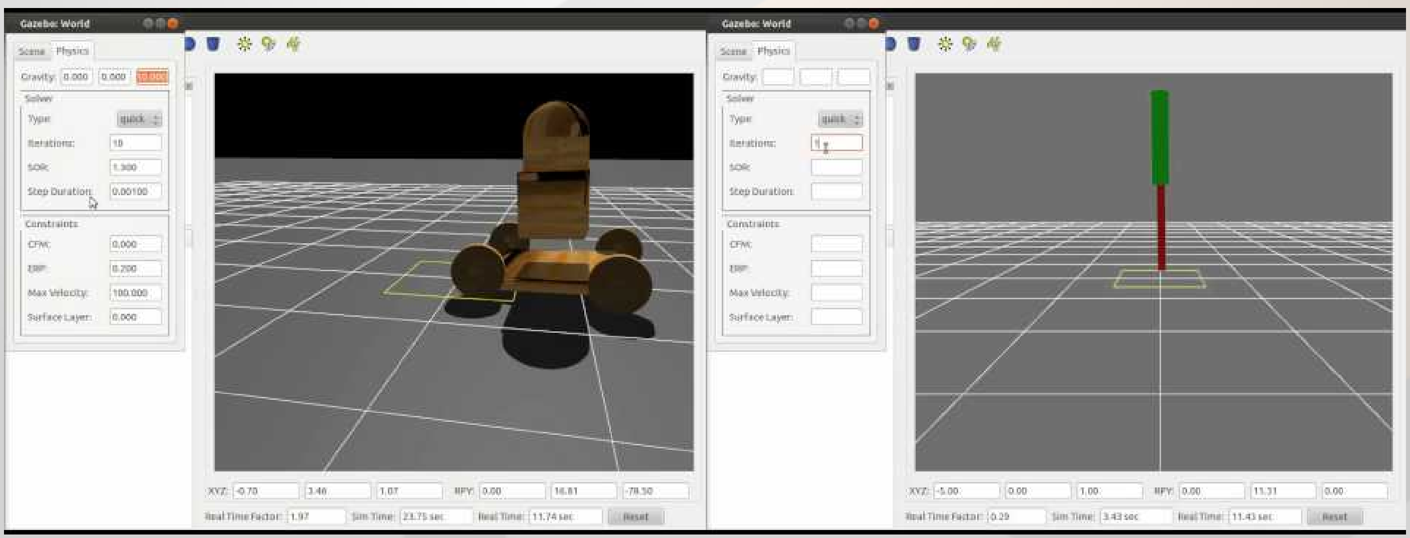
Symptom: Model spins out of control

- **Cause:** Large accelerations (f >> m)
- **Solution**: Remove forces, e.g. disable plugins that sets forces on joints or links, and see if problem goes away
- Look for tiny inertia values.

Troubleshooting Models

Symptom: Model is jittery

- **Cause:** Stiff system. Large mass ratio between connected links
- **Solution**: Reduce time step size or increase inner iteration counts



Interfaces

Plugins

Programmatic interface to Gazebo

Types

- System: Control the load and init process
- World: All models and physics engine
- Model: Joints and links
- Sensor: Control data generation and processing

Use cases

- System: Specify custom search paths
- World: Dynamically change physics engine
- Model: Joint controller, such as a differential drive
- Sensor: Data filtering or add noise models

Creating Plugins

- Reference
  - Gazebo wiki tutorials and API specification
  - Examples distributed with the gazebo sources
- ROS plugins
  - Gazebo ROS package provides interface between Gazebo and ROS framework
  - gazebo_plugins ROS package
- Contribute plugins
  - Submit patches to Gazebo
  - Near future: Online database for plugins

Interprocess Communication

Topics

- Usage nearly identical to ROS
  - PublisherPtr pub = node->Advertise<*msg_type*>*(topic_name*);
  - SubscriberPtr sub = node->Subscribe(*topic_name*, *callback*);

Topics vs plugins

- Topics: Run server remotely, start & stop client
- Plugins: Access to complete API, updates every cycle

Commandline Tools

Gazebo tools

- System inspection: gztopic, gzstats
- Insert and remove models: gzfactory

ROS tools

- rosrun gazebo spawn_model
- rosrun gazebo urdf2model

Getting Help

ROS Answers
- answers.ros.org

Gazebo mailing list
- gazebosim.org/support.html

Wiki and Tutorials
- gazebosim.org/wiki

Contributing code
- Submit patches (kforge.ros.org/gazebo/trac)
- Send email to mailing list for suggestions

Questions

Plugin Examples

Differential Drive

- Controls two joints attached to a chassis and wheels
- Accepts velocity commands, produce joint torques
- Example usage: Pioneer2dx mobile base

ROS PR2 Controller

- gazebo_ros_controller_manager ROS plugin
- Mimics the real PR2 motors at transmission level
- Allows code developed in simulation run on a real PR2

Topic Examples

Graphical Interface

All communication between the server and client is handled via topics

Player Interface

Plugins are loaded into Player which then communicate to Gazebo via Topics

Command line tools

Report statistics and offer basic world control functionality