RCLL Winter School:  
Simulation of the RCLL with Gazebo  

Frederik Zwilling
1 Introduction

2 Gazebo Simulator

3 RCLL Simulation

4 Developing in Gazebo

5 Hands-On

6 Summery
1 Introduction

2 Gazebo Simulator

3 RCLL Simulation

4 Developing in Gazebo

5 Hands-On

6 Summery
Problems with Testing

- Hardware for full field really expensive
- Operational robot hardware required
- Lower level components need to work
- Time consuming setup of tests
- Especially with multiple robots
Simulation at a Glance

- Simulation of the RCLL
- Testing in a virtual environment
- Based on Gazebo Simulator
- 3D Simulation, Physics, Visual
- Multi-Robot
- Refbox controlled game
- Exchanges sensors and actuators with simulated ones

⇒ Close to real testing
Simulation Advantages

- Code and test everywhere
- Cheap, fast, scalable way for testing
- Running whole games with Refbox control
- Useful for integration testing
- Strategy evaluation with automated games
1 Introduction

2 Gazebo Simulator

3 RCLL Simulation

4 Developing in Gazebo

5 Hands-On

6 Summery
http://gazebosim.org/

- 3D Multi-robot simulator
- Physics simulation (ODE,Bullet)
- Open Source
- Widely used
- e.g. used in the Darpa Robotics Challenge
  ⇒ Many existing models/plugins
Gazebo Simulator - Building Blocks

World, Robots, Objects

- Built with Simulation Description Format (SDF)
- XML-like modeling of parts, joints, sensors

Robot Control, Sensor Logic, World Logic

- Plugins written in C++
- Publisher/Subscriber messaging with Protocol Buffers (Protobuf)
Gazebo
- Direct access through pub-sub middleware
- Allows integration with any existing software

Fawkes
- We provide integration modules
- Simulation or real hardware opaque for behavior system

ROS
- Widely used with Gazebo
- Gazebo integration exists for most sensors
1 Introduction

2 Gazebo Simulator

3 RCLL Simulation

4 Developing in Gazebo

5 Hands-On

6 Summery
RCLL Simulation - Features

Actuators
- Robotino 3 (Motor Command)
- Logical Gripper

Sensors
- Laser-/Distance- Sensors
- Cameras
- Ground Truth Localization, Vision (e.g. Light Signal)

Game
- Physics simulation
- Refbox integration
- MPS and Light Signal simulation

Fawkes
- Already integrated (same interface as on real robot)
- Automated Simulation Runs/Competition
Simulation Architecture

Gazebo

RCLL Environment

Models

Gazebo Robot 1
Motor, Laser, Cam, ...

Gazebo Robot 2
Motor, Laser, Cam, ...

Gazebo Robot 3
Motor, Laser, Cam, ...

Gazebo API

Referee Box

Visualization

Robot 1
Fawkes, ROS, ...

Robot 2
Fawkes, ROS, ...

Robot 3
Fawkes, ROS, ...
Multi-Level Abstraction

Higher level abstraction
- Bypass acquisition of sensor data
- Allows to run with fewer functional components

Lower level abstraction
- Generate sensor data from simulation
- Run functional processing components
Introduction

Gazebo Simulator

RCLL Simulation

Developing in Gazebo

Hands-On

Summery
Simulation Description Format (SDF) - World

- **Specification** [http://sdformat.org](http://sdformat.org)
- Load world with `gazebo llsf.world`
- Change simulation speed with step size and update rate
- Include other models with name
- Pose x,y,z,roll,pitch,yaw in parent frame
- Plugin used in World, Model, Sensor

```xml
<sdf version="1.4">
    <world name="LLSF">
        <physics type="ode">
            <max_step_size>0.004</max_step_size>
            <real_time_factor>1</real_time_factor>
            <real_time_update_rate>300</real_time_update_rate>
        </physics>
        <include>
            <uri>model://german_open_field</uri>
        </include>
        <include>
            <name>C-BS</name>
            <uri>model://mps_base</uri>
            <pose>-5 4.9 0 0 0 0</pose>
        </include>
        <plugin name="mps_spawn" filename="libmps_placement.so"/>
    </world>
</sdf>
```
SDF - Model

- Description in `model.config`
- Definition in `model.sdf`
- Collisions for physics, sensors

```xml
<model name="workpiece_base">
  <pose>0 0 0.915 0 0 0</pose>
  <static>false</static>
  <link name="link">
    <collision name="cylinder_collision">
      <geometry>
        <cylinder>
          <radius>0.02</radius>
          <length>0.0225</length>
        </cylinder>
      </geometry>
    </collision>
    <visual name="cylinder_visual">
      <geometry>
        <cylinder>
          <radius>0.02</radius>
          <length>0.0225</length>
        </cylinder>
      </geometry>
    </visual>
  </link>
</model>
```

- Visuals for camera
Define the relative movement between links
Required to fix multiple links/included models together
- Add and configure predefined sensors
- Some publish data automatically (Laser, Camera)
- Otherwise sensor-plugin required to publish data

```xml
<sensor name="laser" type="ray">
    <ray>
        <scan>
            <horizontal>
                <samples>240</samples>
                <resolution>1</resolution>
                <min_angle>-2.094</min_angle>
                <max_angle>2.094</max_angle>
            </horizontal>
        </scan>
        <range>
            <min>0.03</min>
            <max>10.0</max>
            <resolution>0.01</resolution>
        </range>
        <noise>
            <type>gaussian</type>
            <mean>0</mean>
            <stddev>0.005</stddev>
        </noise>
    </ray>
    <always_on>1</always_on>
    <update_rate>15</update_rate>
</sensor>
```
Gazebo-Plugins

- Model control, sensor and game logic
- API: http://gazebosim.org/api.html
- Plugins for Models, Sensors, World

```cpp
class Motor : public ModelPlugin
{
    override Load() and Reset()
    register OnUpdate()

    void Motor::OnUpdate(const common::UpdateInfo & /*_info*/)
    {
        this->update_connection_ = event::Events::ConnectWorldUpdateBegin(
            boost::bind(&Motor::OnUpdate, this, _1));
    }
};
```
Gazebo-Plugins - Messaging

- Communication between Gazebo-plugins with Publisher/Subscriber (also to Fawkes)
- Init communication nodes for world or specific robot

```cpp
node_ = transport::NodePtr(new transport::Node());
node_->Init(model_->GetWorld()->GetName()+"/"+name_);
```

- Publisher with topic name (~ depends on init)

```cpp
localization_publisher_ = node_->Advertise<msgs::Pose>("~/gazsim/localization/");
```

- Subscriber and callback function

```cpp
localization_subscriber_ = node_->Subscribe("~/gazsim/localization/", &Mps::on_puck_msg, this));
void Mps::on_puck_msg(ConstPosePtr &msg)
```

- gz topic useful for debugging
Gazebo-Plugins - Messaging - Protobuf

- Protobuf messages as in Refbox

## Protobuf Message

```protobuf
package gazebo.msgs;

import "vector3d.proto";
import "quaternion.proto";

message Pose
{
  optional string name = 1;
  optional uint32 id = 2;
  required Vector3d position = 3;
  required Quaternion orientation = 4;
}
```

## Usage in Gazebo-Plugin

```c++
//Build message
msgs::Pose posMsg;
posMsg.set_name(name_);
posMsg.mutable_position()->set_x(model_->GetWorldPose().pos.x);
posMsg.mutable_position()->set_y(model_->GetWorldPose().pos.y);
posMsg.mutable_position()->set_z(model_->GetWorldPose().pos.z);

//send
localization_publisher_->Publish(posMsg);
```
Gazebo-Plugins - Hints

Control of simulation with API
- Usage of World, and Model functions

Control of simulation with messages
- Change visuals by sending Visual msg on ~/visual topic
  Name has to match
  (e.g. M-BS::light_signals::link::redon)
- Spawn models by sending Factory msg on ~/factory topic
- Control joints by sending Joint msg on ~/joint topic

Slow Setup
- Models spawned or modified to early appear at origin
- Initial timeout sufficient

More Help
- http://gazebosim.org/tutorials
- http://answers.gazebosim.org/questions/
1. Introduction
2. Gazebo Simulator
3. RCLL Simulation
4. Developing in Gazebo
5. Hands-On
6. Summery
Hands-On

Get familiar with Gazebo and the RCLL simulation

- Start Gazebo with the RCLL world
- Start a game in the Refbox
- Spawn and move some models

Add a new sensor to detect workpieces in the gripper

- Create a new model
- Add a ray sensor
  (similar to infrared-sensor)
- Attach the sensor-model to the robot
- Inspect the generated data with `gz topic`
Hands-On

Write a Model plugin for the sensor-model

- Subscribe to the ray-sensor data
- Decide if there is a puck in the gripper
- Send detection result on a new topic (e.g. with a new Protobuf message)

Build your own robot

- Start with the `carologistics-robotino` model and modify it
- Add the robot to the world file
Conclusion and Questions

- Gazebo based simulation 3D, physics, visuals
- Many pre-built models
- Refbox controlled game
- Multi-level abstraction simulate data or information

- SDF for modeling world, robots, sensors, objects
- Gazebo plugins for control, sensor and world logic
- Publisher/subscriber API for communication

Using the Gazebo RCLL simulation is valuable for tests and accelerates the development.