ARGoS User Manual

Carlo Pinciroli

January 30, 2012

Contents

Part I

Using ARGoS

Chapter 1

Introduction

1.1 Where to Download

1.2 Installation

1.2.1 Ubuntu

Required packages for a binary installation: libqt4-opengl, freeglut3 (including all their dependencies) Optional packages for a binary installation: povray

1.3 Your First Experiment

1.4 Main Concepts

- Multiple physics engines
- Threads

Chapter 2

Writing Controllers

Chapter 3

The XML Configuration File

To run a simulation, the user must provide a correct XML configuration file. Such file serves diverse purposes: describing the static layout of the arena, placing the robots, declaring which controllers to use, and more. In the following we will see how this file is organized.

3.1 The General Structure

The XML configuration file of ARGoS is composed of 5 required section and 2 optional ones:

`framework` **(req):** here you set some internal parameters of ARGoS;

`controllers` **(req):** a list of the controllers to use in the experiment, along with a specification of their configuration;

`arena` **(req):** contains the objects that populate the simulated space (i.e., their position, orientation, ...);

`physics_engines` **(req):** a list of the physics engines to use in the environment;

`arena_physics` **(req):** the mapping between objects in the simulated space and the physics engines. Mobile entities can only belong to one physics engine, while immobile entities usually belong to multiple engines;

`visualization` **(opt):** if this section is present, the selected visualization is loaded. Otherwise, ARGoS runs silently.

`loop_functions` **(opt):** if this section is present, the user-defined that contains the dynamic aspects of the experiment is loaded.

3.1.1 The `framework` Section

This section has two tags: `system` and `experiment`. Let's see an example:

```
<framework>
  <system threads="2" />
  <experiment length="70"
              ticks_per_second="10"
              random_seed="124" />
</framework>
```

The `system` tag currently contains only one attribute, `threads`, that stores the number of slave threads used by ARGoS for the computation. If you set it to zero, no threads are employed. If you set it to a number $N > 0$, N slave threads are used. Performance evaluation shows that it makes sense to use threads only when the number of robots to simulate is larger than about 100. Below this threshold, the cost of managing the threads exceeds the benefits. In case you want to use threads, for maximum performance it is better to set the number of threads equal to the number of cores present in your system.

The `experiment` section is used to specify some general aspects of the simulation. The `length` attribute contains the desired lenght of the experiments in seconds. If you set it to zero, the experiment runs infinitely, or until another custom ending condition is met as specified in the `IsExperimentFinished()` method in the loop functions (for more information, see Chapter 4). The `ticks_per_second` attribute specifies how many control steps (ticks) are executed per second. This value is usually set to 10, which corresponds to 100ms-long control steps. Finally, if you set the `random_seed` attribute, the simulation will be repeatable. Noise will be calculated on the basis of the random seed you set. If you don't set it or choose zero as value, the internal computer clock is used to find a random seed and a warning is displayed at run-time.

3.1.2 The `controllers` Section

In the `controllers` section you specify the robot controllers to use for the experiment:

```
<controllers>

  <my_controller_1 id="c1" library="/path/to/libmy_controller_1.so">
    <actuators>
      ...
    </actuators>
    <sensors>
      ...
    </sensors>
    <parameters>
      ...
    </parameters>
  </my_controller_1>

  <my_controller_2 id="c2" library="/path/to/libmy_controller_2.so">
    ...
  </my_controller_2>

</controllers>
```

In Chapter 2 we said that controllers are registered in the system through the `REGISTER_CONTROLLER()` macro. The string identifier you specify as parameter to the macro is used in the XML file as tag (here, `my_controller_1` and `my_controller_2`).

The configuration of each controller is performed specifying the following information:

- a unique identifier, used in the `arena` section to bind robots to controllers (attribute `id`);
- the complete path to the plug-in library file (attribute `library`);
- which actuators and sensors to use, along with their configuration. As shown in the above example, a sub-section is dedicated to each (`actuators` and `sensors`);
- each robot controller accepts an XML sub-tree that is (optionally) parsed by the `Init()` method. The passed tree is the `parameters` sub-section.

3.1.3 The `arena` Section

3.1.4 The `physics_engines` Section

3.1.5 The `arena_physics` Section

3.1.6 The `visualization` Section

3.1.7 The `loop_functions` Section

Chapter 4

The Loop Functions

Part II

Extending ARGoS

Chapter 5

The ARGoS Architecture

Chapter 6

Writing a New Sensor/Actuator

Chapter 7

Writing a New Visualization

Chapter 8

Writing a New Physics Engine