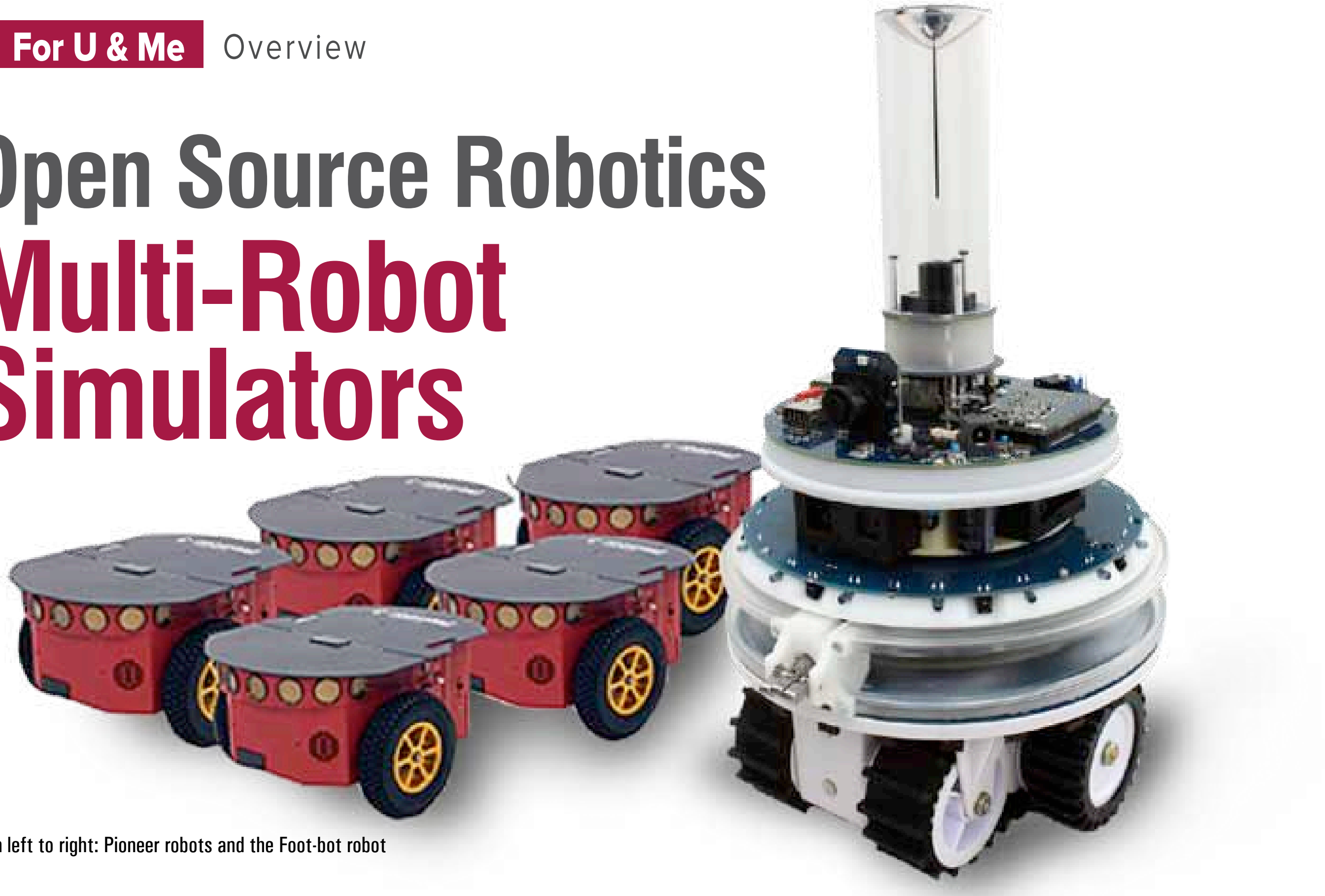
Open Source Robotics
Multi-Robot Simulators


From left to right: Pioneer robots and the Foot-bot robot

This is the second article in a series that focuses on open source software for robotics. This part introduces the Linux enthusiast to multi-robot simulators, by taking a peek into Stage and ARGoS.

Part—2

A robot enthusiast's flight of fancy could envisage a futuristic society where robots and human beings coexist. This apparent utopia would be cohabited by human beings and robots that obey human beings and help in their daily chores, do the jobs that are redundant, come to rescue when human lives are in jeopardy, and are intelligent enough to protect the human populace.

The development of paradigms for a human-robot society led to science fiction writer Isaac Asimov coining the Three Laws of Robotics:

I. A robot may not injure a human being or, through inaction, allow a human being to come to harm.
II. A robot must obey the orders given to it by human beings, except where such orders would conflict with the First Law.
III. A robot must protect its own existence, as long as such protection does not conflict with the First or Second Laws.

These laws have come to form the basis of a stable society containing robots that are subservient to the human population.

Multi-robot systems

To make prophecies about such futuristic societies, it is important to study multi-robot interactions. It is easy to see that a multi-robot team will have clear advantages over a single robot system. Also, improved performance due to distributed sensing, enhanced fault tolerance and parallelism, add to the virtues of a robotic team.

Like many other emerging fields of robotics, algorithms for multi-robot systems are inspired by nature. Foraging for food, division of labour, nest-building, cumulative defence against enemies, leader following, flocking, etc, are common in human, insect and animal societies.

A unique feature of multi-robot systems is that one can distinguish between individual-level behaviour and team-level behaviour. The former refers to what an individual in a team does. The latter refers to the behaviour of the team as a whole. For instance, in schools of fish, the individual behaviour of each fish is to mainly avoid collision with neighbours, and match their attitude. The team-level behaviour, on the other hand, is a remarkable display of cohesive and coordinated motion.

The study of the relationship between individual and team-level behaviour is just beginning. It's a long road to constructing multi-robot systems that are able to display the same complexity and performance of natural systems like insect societies. Also, the cost for this type of research is prohibitive. Robots are expensive, and producing large quantities of them is out of the reach of today's research budget. Luckily, scalable multi-robot simulation programs have come to the rescue.

Robotics: Real robots vs simulations

There exists a wide variety of robot simulators. Those designed before the year 2000 were typically targeted at specific use-cases. It was only during the last decade that PCs started to become powerful enough to allow for more complex designs. Today, there are a few general robot simulators that meet the needs of the robotics community.

The performance of a robot simulator and its accuracy are typically the result of a trade-off between the number of robots and speed of simulation. Performance evaluation of *Stage* and *ARGoS* discussed in later sections illustrate this trade-off. When the aim is to simulate large teams of robots, this trade-off can become dramatic. For this reason, in the last few years, a few simulators have been designed to provide good performance for large teams of robots.

The *Stage* simulator

Stage is a part of the Player Project (Player/Stage/Gazebo) and is now also a part of ROS (Robot Operating System); there have been significant changes since version 3 to improve its simulation performance. Maybe the most important novelty is the support for multi-threaded execution.

Stage is supported in nearly all UNIX-based OSs and in Mac OS X. It is interesting to know that the name *'Player'* and *'Stage'* were inspired by the quote, *"All the world's a stage,"* by William Shakespeare in *As You Like It*.

Stage provides the user with a two-dimensional graphic environment, which has facilities for a perspective camera, making it effectively a 2.5 dimensional simulator. *Stage* has facilities for modelling the robot and its sensors using simple scripts. Some in-built *Stage* controllers are the *'wander'* controller, which enables a wandering kind of behaviour; the *'lasernoise'* controller, which helps in studying the noise generated in the laser sensor of the robot; and the *'pioneer_flocking'* controller, which enables flocking behaviour for a number of pioneer robots.

To implement the *'pioneer_flocking'* controller, *cd* to *Stage-source/worlds* and run the world file with the following command:

```
stage pioneer_flocking.world
```

That will pop up a screen similar to the one in Figure 1.

Stage shines at simulating navigation- and sensing-based experiments. It can simulate one robot about 1000 times faster than real-time, and 1000 robots in about real-time. However, the kinematic physics models employed in *Stage* prevent one from executing experiments that involve pushing and pulling objects, as well as robot self-assembly.

To cope with this issue, the Player Project offers Gazebo, which is a three-dimensional dynamics simulator. In Gazebo,

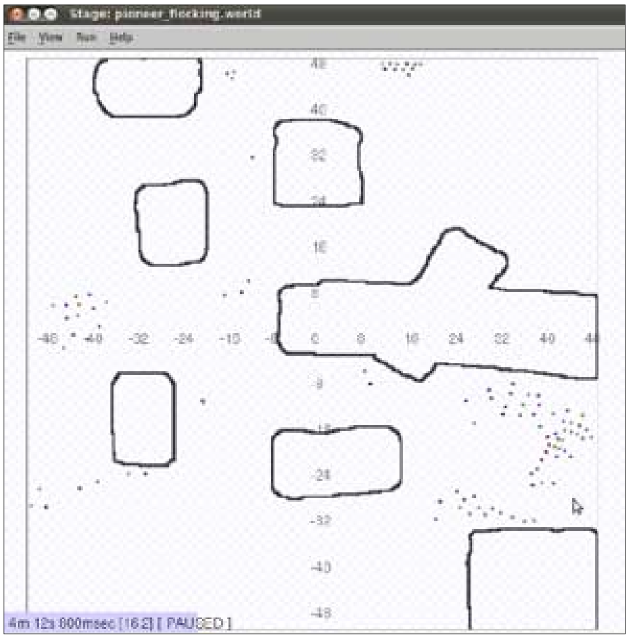

Stage: pioneer_flocking.world
File View Run Help
4m 12s 800msec [16.2] [PAUSED]

Figure 1: Multi-coloured pioneer robot models forming swarms

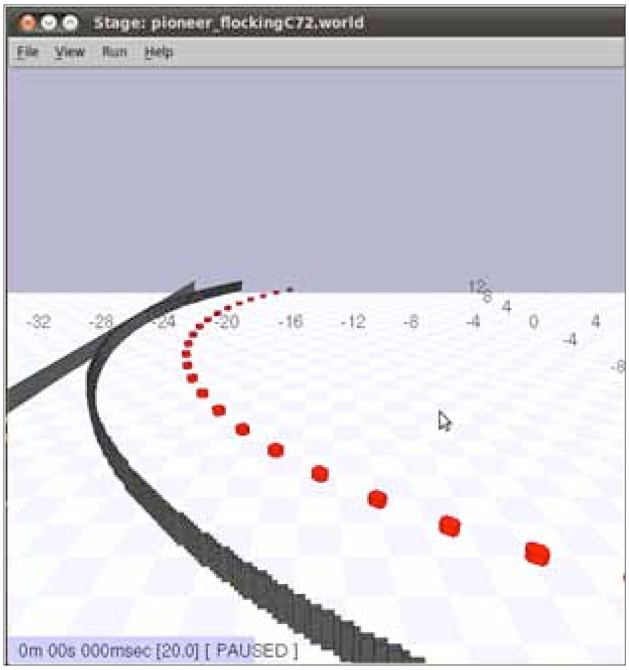

Stage: pioneer_flockingC72.world
File View Run Help
0m 00s 000msec [20.0] [PAUSED]

Figure 2: Pioneer robots formation

one can run very complex experiments. However, the improved accuracy of the robot models makes this simulator much slower than *Stage*.

Stage is also a part of ROS. However, at the time of writing, the latest ROS release *Electric Emys* does not support *Stage* controllers through the ROS nodes.

The stable release of *Stage* can be downloaded from *playerstage.sourceforge.net/*. Alternatively, if you feel adventurous and want to try the bleeding edge, you can download the development version at *https://github.com/rtv/Stage*.

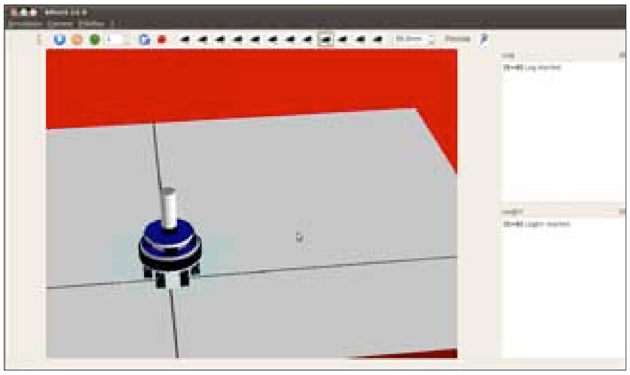


Figure 3: ARGoS basic simulation

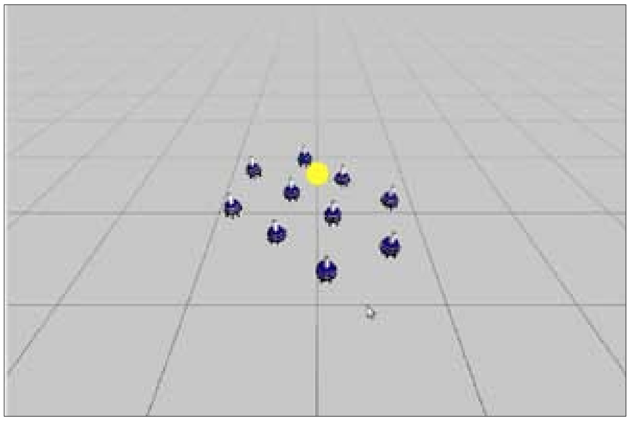


Figure 4: Flock formation in ARGoS

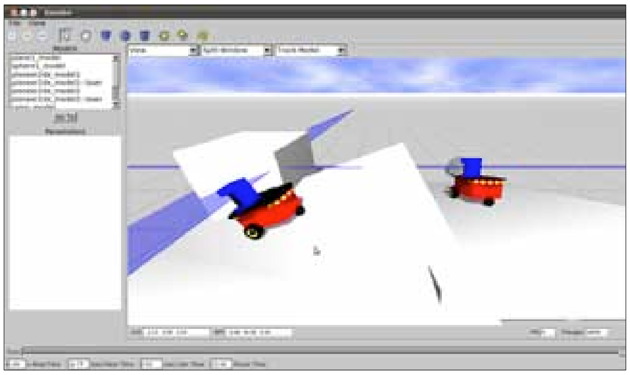


Figure 5: Two pioneer robots in Gazebo

The *ARGoS* simulator

ARGoS (Autonomous Robots Go Swarming) is a multi-robot simulator designed to support large teams of robots. Its design is pretty different from the design of other simulators. Its most distinctive feature is that the 3D simulated world can be divided in regions, and each region can be assigned to a different physics engine. Furthermore, *ARGoS*' design revolves around the concept of tunable accuracy. In other words, in *ARGoS*, everything is a plug-in (robot models, sensors, actuators, physics engines, visualisations, etc) and the user can select which plug-ins to use for an experiment. Since different plug-ins have different accuracy and computational costs, users can choose which plug-ins to use for each aspect of the simulation and assign resources only where it matters. This makes the simulation as fast as possible.

At the time of writing, *ARGoS* supports the Swarmanoid robots (foot-bot and eye-bot) and the e-puck. *ARGoS* supports Linux and Mac OSX. Binary packages are available for Ubuntu, Slackware and Mac OS-X. In addition, a generic binary package can be used for other Linux distributions. Everything is downloadable from *iridia.ulb.ac.be/argos/download.php*.

Robot control code for *ARGoS* is written in C++. Experiments are configured through an XML file. To run a demo simulation in *ARGoS*, download the examples from the same URL, uncompress the archive, and run the experiment with the following command:

```
launch_argos -c xml/diffusion_1.xml
```

That results in a screen popping up, as shown in Figure 3, with a single foot-bot robot in a red box-like environment.

Compared to Stage, *ARGoS* provides a 3D simulation environment. In addition, since the physics engines can be chosen by the user, any kind of experiment is possible, including complex self-assembly. Its performance is found to be superior to *Stage*'s. With the full power of four cores on a normal desktop PC, *ARGoS* can simulate more than 4000 robots in real-time.

Watch out for Part 3 of this series, which will contain a comprehensive discussion on ROS. END

Reference

[1] I Robot, 2004 movie, *www.imdb.com/title/tt0343818/*
[2] R Vaughan, 'Massively Multiple Robot Simulations in Stage,' Swarm Intelligence 2(2-4):189-208, 2008. Springer.
[3] N Koenig and A Howard, 'Design and Use Paradigms for Gazebo, An Open-Source Multi-Robot Simulator,' a paper presented at the IEEE/RSJ International Conference on Intelligent Robots and Systems, 2004
[4] Carlo Pinciroli, Vito Trianni, Rehan O'Grady, Giovanni Pini, Arne Brutschy, Manuele Brambilla, Nithin Mathews, Eliseo Ferrante, Gianni Di Caro, Frederick Ducatelle, Timothy Stirling, Álvaro Gutiérrez, Luca Maria Gambardella and Marco Dorigo. 'ARGoS: a Modular, Multi-Engine Simulator for Heterogeneous Swarm Robotics.' – proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2011), pages 5027-5034, IEEE Computer Society Press, Los Alamitos, CA.

By: A Bhaumik

The author is a faculty member at the Birla Institute of Technology, Mesra, Ranchi. He is a robotics enthusiast and his ramblings on mobile robots can be found at his blog, *mobotica.blogspot.in/*. He can be reached at *arkapravobhaumik@gmail.com.*