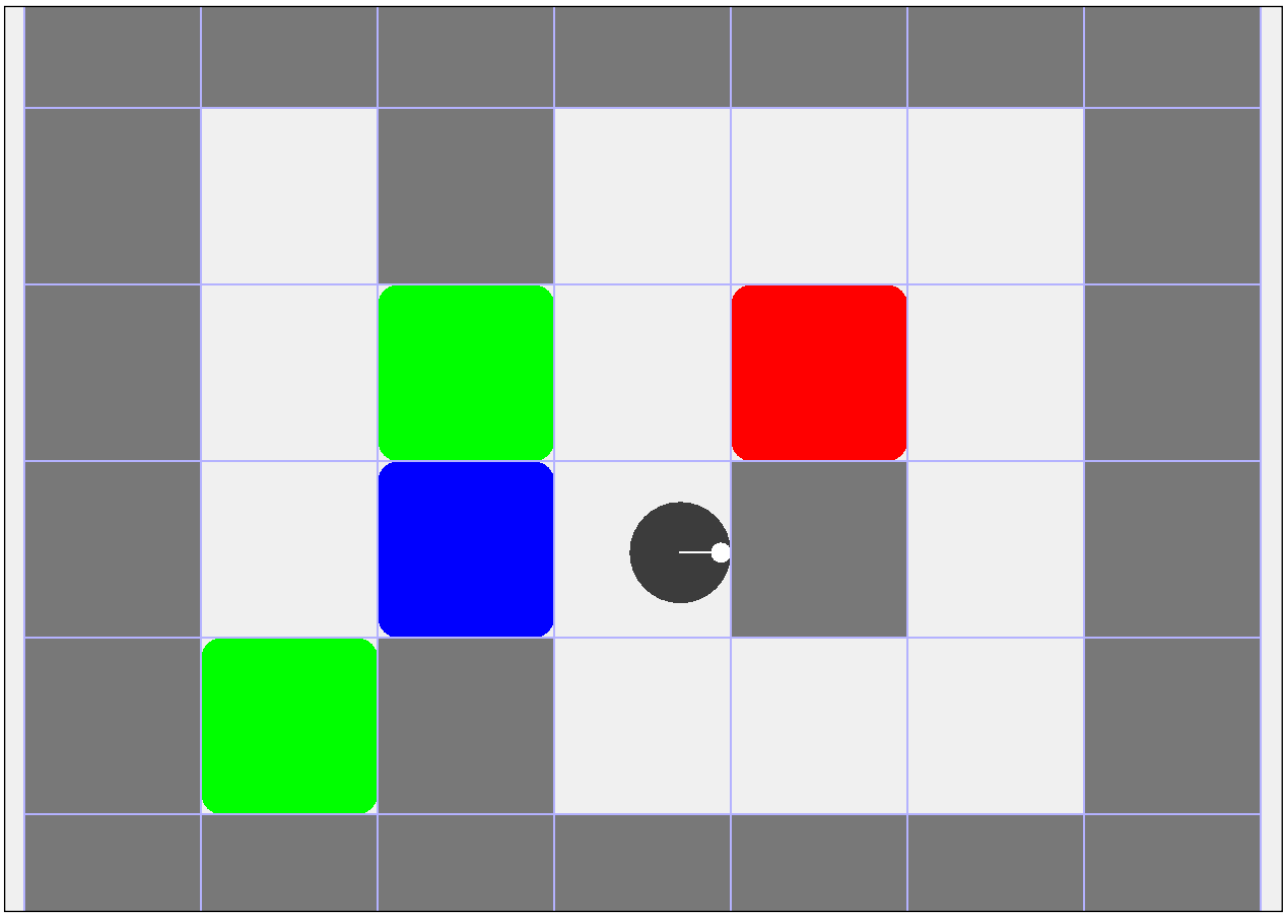
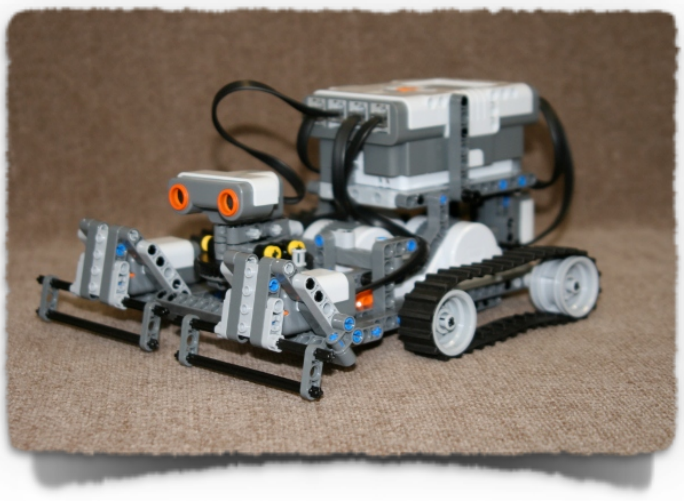
COMP329RoboSim

Simple Robot Simulator for COMP329 Assignments

Dr Terry R. Payne - 15 December 2016

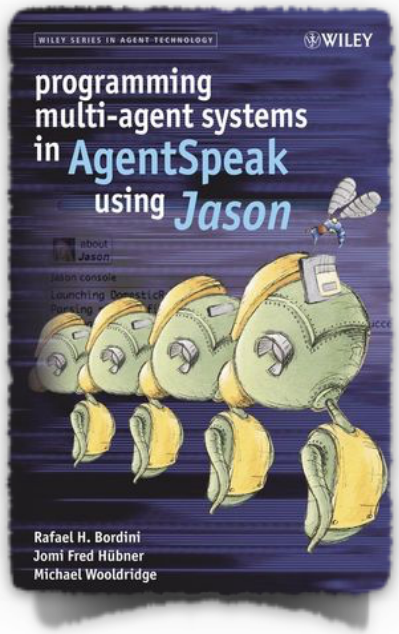




Introduction

The ***RoboSim*** package is designed to simulate a simple LeJOS style Lego Robot environment that can be used to test out simple programs, and to simulate robot control within the AgentSpeak / Jason environment. It has been modelled around the Lego Robot used for the first Assignment within ***COMP329 Robots and Autonomous Systems*** and provides complementary functionality including:

- Bump Sensor (in the direction of travel)
- Directional UltraSound Proximity Sensor
- Colour Sensor
- Pose Query methods
- Speed / Rotation / Distance methods
- Configurable arena modelling
- Obstacles and "victims"
- Real Time Robot Diagnostic Information
- Dead Reckoning control[1]



The functionality provided within the RoboSim API is basic, but provides the necessary action and perception methods that more complex functionality can build upon. This guide is design to get you started in using the RoboSym package within JASON running under Eclipse, and provides an overview of the API itself, complete with example code fragments. A number of assumptions have been made with the package, including:

- The robot has a 10cm radius circular body to simplify navigation around obstacles
- The robot can navigate around cells and can be positioned independently of the number of cells in the arena, and can move in any direction
- Cells are 35cm (350mm) square; this is also the size of obstacles and victims
- The robot can rotate in a positive and negative direction by specifying the number of degrees to rotate. Simple rotation (e.g. turning left or right) can be achieved by specifying an angle of 90 (or -90), but finer grain control is also possible
- The robot can move a fixed distance and will then stop, based on a distance to travel. This is specified in mm, and thus moving from the centre of one cell to the centre of another can be achieved by moving 350mm.

[1] Uncertainty will be included in future versions, but for the sake of simplicity we relax the need for sophisticated localisation techniques for the sake of the current Assignment 2.

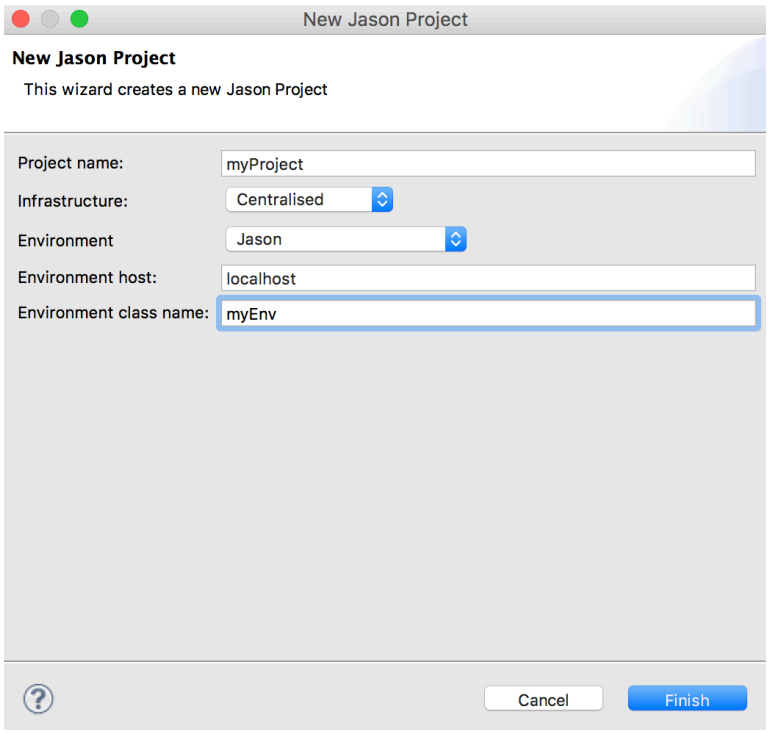
- The bump sensor triggers if the body comes into contact with an obstacle that causes the robot to stop. If the robot rotates such that it no longer collides with the obstacle, then the sensor resets.
- Methods for moving forward, rotating or turning the UltraSound Sensor are blocking (i.e. the code will wait until the robot has completed the task), but the robot will move at a specified speed until it has reached its goal.

Note that the package is a prototype and still in development. If you find any problems, or issues, or it behaves unexpectedly, then contact me (trp@liv.ac.uk) providing details of the code and issue found.

Adding the RoboSym package to a new JASON project

The following assumes that you already have JASON installed on a PC (MacOSX, Windows or Linux) running under Eclipse. The jar file, **comp329robosim.jar**, is available from the Module Website, which also includes links to the JASON SourceForge site.

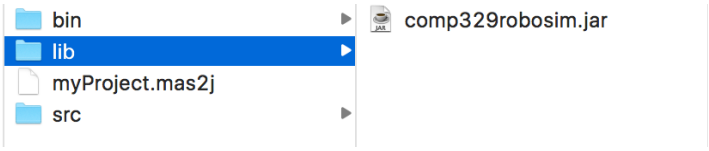
Create a JASON project with an environment, and give the project a name and also name the environment. This will create a new project in your workspace, with a number of initial files. In the example opposite, I have created a project called **myProject** with an environment class called **myEnv**.

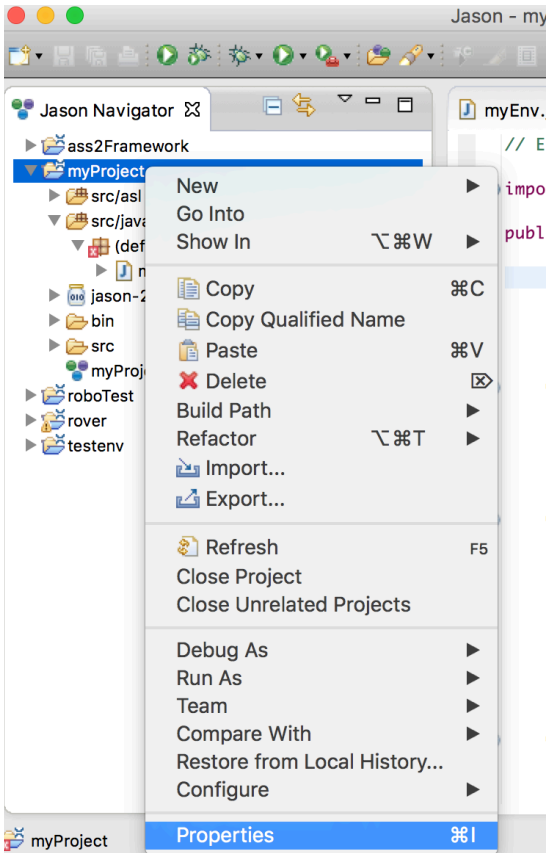


If you try to build and run this, you will find errors in the myEnv.java file that are due to an unhandled exception. For now, comment out the offending line (**`addPercept(ASSyntax.parseLiteral("percept(demo)"));`**) and check that you can now run the project, which should result in one agent saying "Hello World". The next step is to:

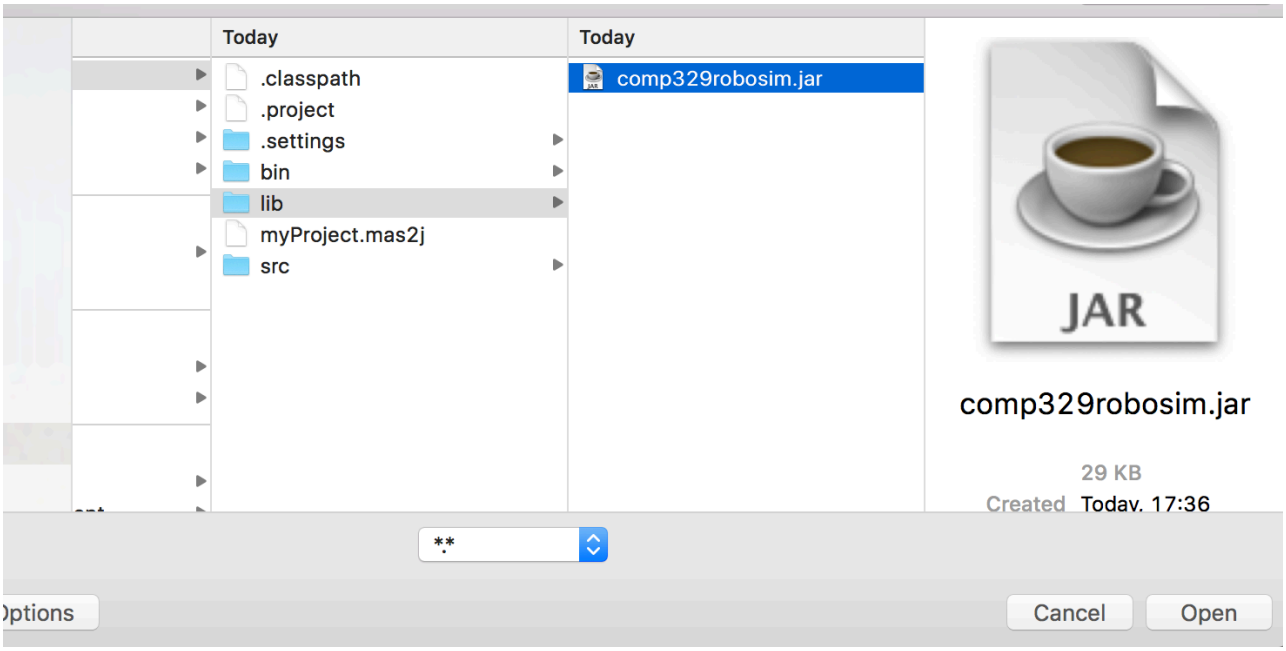
1. Tell Eclipse about the jar file
2. Tell Jason about the jar file

If you close Eclipse and navigate to the directory containing your project you will see two directories: **src** and **bin**, and the **myProject.mas2j** file.Create a new directory called **lib**, and copy the jar file into that **lib** directory:





The Jason environment will now be able to find the jar file when it compiles. However, you will still need to add the jar file to the Eclipse project. Right-click on the project within Eclipse to bring up a menu and select **Properties** to open up the project properties. In the left hand panel you will see the option **Java Build Path** - this is where you inform Eclipse of resources that you can include into your project. Select this, and then select the button **Add External JARs** to add the jar file to your project. A file browser will appear - if you don't see the relevant directories, you can toggle the filter between jar and */* files. Select



the jar file in the lib directory and then select Open. Finally, select OK to finish adding the jar file. This will add the jar file to your project and should list it in the project resources in Eclipse. You are now ready to import the **comp329robosim** package classes in your project.

You will need to create a new object of of the **EnvController** class, which configures the simulator. You can also request from this object a reference to **RobotMonitor** Object, which will provide you control of your robot.

First, expand the jar resource to see its classes. You will see a file **DefaultConfig.txt** file. Create a new file somewhere in your file directory, and copy over the contents - this will be your configuration file. For the final evaluation, we will give you a new configuration file that defines the final location of obstacles and victims. Note the path to the location of this file, as you will need to specify it when creating the **EnvController** class instance.

```
// Environment code for project myProject

import jason.asSyntax.*;
import jason.environment.*;

import java.util.logging.*;

import comp329robosim.EnvController;
import comp329robosim.RobotMonitor;

public class myEnv extends Environment {

    //EnvController controller;

    private Logger logger = Logger.getLogger("myProject."+myEnv.class.getName());

    // ========= Instance Variables for Robosym ========
    EnvController controller;
    RobotMonitor myRobot;
    // ================================================

    /** Called before the MAS execution with the args informed in .mas2j */
    @Override
    public void init(String[] args) {
        super.init(args);
        // addPercept(ASSyntax.parseLiteral("percept(demo)"));

        // ========= Creating the Robosym Objects ========
        controller = new EnvController("/Users/trp/Desktop/robosimDefaultConfig.txt", 7, 6);
        myRobot = controller.getMyRobot();
```

It is up to you have you manage your objects in your project, but to get started, create two instance variables in your **myEnv** class for the **EnvController** and **RobotMonitor** objects. Then in the init method, instantiate these. Ensure that you import the classes from the jar file. Note that the line below takes three arguments; the first is the path to the config file (you will probably need to change this to the location of your file) and the second two define the size of the arena (including the bounding objects).

```
controller = new EnvController(
        "/Users/trp/Desktop/robosimDefaultConfig.txt", 7, 6);
```

If you then run the project, you should see the map appear, with a circular robot in the top left hand corner. The line in the robot indicates the heading, whereas the small circle indicates the direction of the UltraSound Sensor.

If you are having problems getting this far, then contact Jeffrey or myself to get advice on how to fix them. You need to get to this point before you can start controlling the robot.

More details will follow soon, including code fragments on the different Robot methods. For now, a javadoc archive is also available from the module website which contains details on the different methods. To get you started, try the following lines in your code..

```
myRobot.monitorRobotStatus(true);
myRobot.setTravelSpeed(100);        // 10cm per sec
myRobot.setDirection(0);

while (myRobot.getUSenseRange() > 700) {
        myRobot.travel(350);        // travel 35 cm
}
myRobot.rotate(90);                 // Turn Left

while (!myRobot.isBumperPressed()){
        myRobot.travel(350);        // travel 35 cm
        if (myRobot.getCSenseColor().equals(Color.GREEN)) {
                System.out.println("found Green cell at location ("+
                                myRobot.getX()+","+
                                myRobot.getY()+") with heading "+
                                myRobot.getHeading());
        }
}
```

See the following page for a screenshot!

```
import jason.asSyntax.*;
import jason.environment.*;

import java.awt.Color;
import java.util.logging.*;

import comp329robosim.EnvController;
import comp329robosim.RobotMonitor;

public class myEnv extends Environment {

    //EnvController controller;

    private Logger logger = Logger.getLogger("myProject."+myEnv.class.getName());

    // ========= Instance Variables for Robosym ========
    EnvController controller;
    RobotMonitor myRobot;
    // =================================================

    /** Called before the MAS execution with the args informed in .mas2j */
    @Override
    public void init(String[] args) {
        super.init(args);
        // addPercept(ASSyntax.parseLiteral("percept(demo)"));

        // ========= Creating the Robosym Objects ========
        controller = new EnvController("/Users/trp/Desktop/robosimDefaultConfig.txt", 7, 6);
        myRobot = controller.getMyRobot();

        myRobot.monitorRobotStatus(true);
        myRobot.setTravelSpeed(100);// 10cm per sec

        myRobot.setDirection(0);

        while (myRobot.getUSenseRange() > 700) {
            myRobot.travel(350);        // travel 35 cm
        }
        myRobot.rotate(90);             // Turn left

        while (!myRobot.isBumperPressed()){
            myRobot.travel(350);        // travel 35 cm
            if (myRobot.getCSenseColor().equals(Color.GREEN)) {
                System.out.println("found Green cell at location ("+
                        myRobot.getX()+","+
                        myRobot.getY()+") with heading "+
                        myRobot.getHeading());
            }
        }

    }

    @Override
```