Moose's Software Valley

Robot Simulator

Copyright (C) Mike "Moose" O'Malley http://move.to/moose

This program is Freeware – see "Freeware Information" below.

All trademarks are the properties of their respective owners.

TABLE OF CONTENTS

Introduction 1
The Issues, Hassles, Pain, Expense of Real "Battery Controlled" Robots 2
Installation Instructions 3
Uninstall Instructions 3
Minimum System Requirements 3
Recommended System Requirements 3
How to Use This Program 4
Available Robots 5
RoboSlang - Introduction 6
Calibrating your Robot 9
Future Changes / Still to Do / Work Outstanding 9
Reviews / Awards / CDs 10
Freeware Information 10
Warranty 10
Amendment History 10

Introduction

Robot Simulator is for Windows 95, 98, ME, NT, 2000, XP, or later.

This program is free software. Anyone - any person, any company, or any business - can use this program for free. No fees or payment is required. See "Freeware Information" below.

This Robot Simulator program allows you to simulate and control your own virtual robot(s) on screen.

I got the idea for developing this program after attending a Lego Mindstorms Programming workshop.

Robot Simulator was developed to allow anyone to control and program robots without the expense, hassles, pain, or flat batteries of using real battery controlled robots.

Typically, each robot consists of two (or more) motors and one (or more) sensors.

In this program, the basic robots have 2 wheels and 2 light sensors, and more advanced robots (included with this program) currently have anything up to 5 sensors (and more robots will be added in the future, as and when needed, depending on support / feedback received).

Motors (connected to wheels) are on the left and right side of the robot, and these can independently control the motion of the robot - that is, they can be moving the wheels on each side at different speeds and in forward or reverse directions. Motor speeds range from -10 to 10 : 1 is slow forwards, 10 is fast forwards, and -10 is fast backwards.

You can also control the robot manually - using the controls on the top right corner of the screen - by setting the angle and moving the sliders to change the speeds of the motors on each side of the robot.

And, you can re-position the robot at any time by left clicking the mouse, and you can bookmark (and restore) its position and orientation by using the buttons in the top right corner of the screen.

Or (and this is where the real fun and purpose of the program really is), you can write a script to program the robot to act on its own - this script language is called "**RoboSlang**", and this script runs inside a special little interpreter that I have designed and developed especially for this project.

A selection of sample scripts are provided with the program to help get you started.

You can change scripts, backgrounds, etc – simply use the buttons provided on the main screen.

You can change robots as well.

Robot Simulator was developed using **Borland Delphi v5.x**.

Robot Simulator is for Windows 95, 98, ME, NT, 2000, XP, or later. This program will **NOT** run under Windows 3.x (even with Win32 and WinG installed).

The Issues, Hassles, Pain, Expense of Real "Battery Controlled" Robots

The Lego Mindstorms and other real robots are great fun to mess around with however, there were some issues :

- You have to download the programs to the robot via a wireless LAN - which is quite fast and easy, but, after a lot of repetitions, it does become a bit of a pain.
- You cannot try out your programs on a virtual robot and see if they worked before downloading them to your real, physical robot.
- The robots chew through brand new sets batteries within a few hours of very occasional use, so they are expensive to keep running. Re-chargeable batteries are a must – but these are expensive. Batteries, in general, are bad for the environment when disposed of.
- If the batteries went completely flat, then you have to re-download the Operating System (OS) to the robot and this took about 5 minutes.
- The robots are NOT cheap to buy. (They are not that expensive either really …)
- The robots - like all machines - will eventually wear out.
- The robot kits are pretty limited.
- The left and right motors were called "A" and "C" and the left, front, and right sensors were called "1", "2", and "3" respectively - which is hardly intuitive.
- Additional hardware - sensors, motors, etc are expensive.
- Accessories - like playing fields for Soccer and Rescue are pretty expensive to buy.
- You cannot debug your programs interactively or watch the programs actually running on the robot. So, if things didn't work as expected, it is sometimes hard to determine why.
- Programming the robots involved using a language of icons and symbols, which is fairly well implemented, but these icons then had to be "wired" together (using a roll of wire) to indicate the sequence, and this was a major pain.
- To get the programming icons to modify their behaviour (e.g. nominate which sensor to read, or how many times to loop, etc), you had to hang boxes containing variables off each icon - which looked messy and was hardly intuitive.

- The programming language cannot be read or understood until you understand what dozens of icons mean / achieve. As with all graphical icons, the meaning of some is obvious, but most are obscure.
- Anything other than basic dance routines or very basic edge following routines are difficult to program - and many screen fulls of icons (linked together by wire) may be required to achieve something more complex - like a proper line follower robot that takes shortcuts.
- The graphical programming language cannot easily be extended or customised.
- Robots cannot be made to teleport, or safely / legally made to fire military grade lasers or missiles, etc. ;) But virtual robots can !!

Robot Simulator overcomes most (if not all) of these limitations.

Installation Instructions

Download the "Robot_Simulator.ZIP" file from my web page, extract the contents to a directory (such as c:\Robot\), and execute the program by double clicking on **Robot_Simulator.EXE**.

N.B. You need to have a recent version of **Adobe Acrobat Reader** or a similar program installed on your computer to enable you to view this PDF document. Adobe Acrobat Reader can be downloaded from here :

http://www.adobe.com/support/downloads/main.html

Uninstall Instructions

Alternatively, use Windows Explorer to navigate to where you installed the Robot Simulator (such as c:\Robot\) and delete all files in this directory.

Minimum System Requirements

- Pentium II 500 MHz or higher
- 16 MB of RAM or more
- Windows 95 or later
- Screen resolution of 800x600 in 15 bit colour (32,000 colours) or better.

Recommended System Requirements

- Pentium II 500 MHz or higher
- A graphics card – rather than on-board video
- 128 MB of RAM or more
- Windows XP or later
- Screen resolution of 800x600 in 16 bit colour (64,000 colours) or better.

How to Use This Program

Robot Simulator is a program that allows you to simulate and control your own virtual robot(s) on screen.

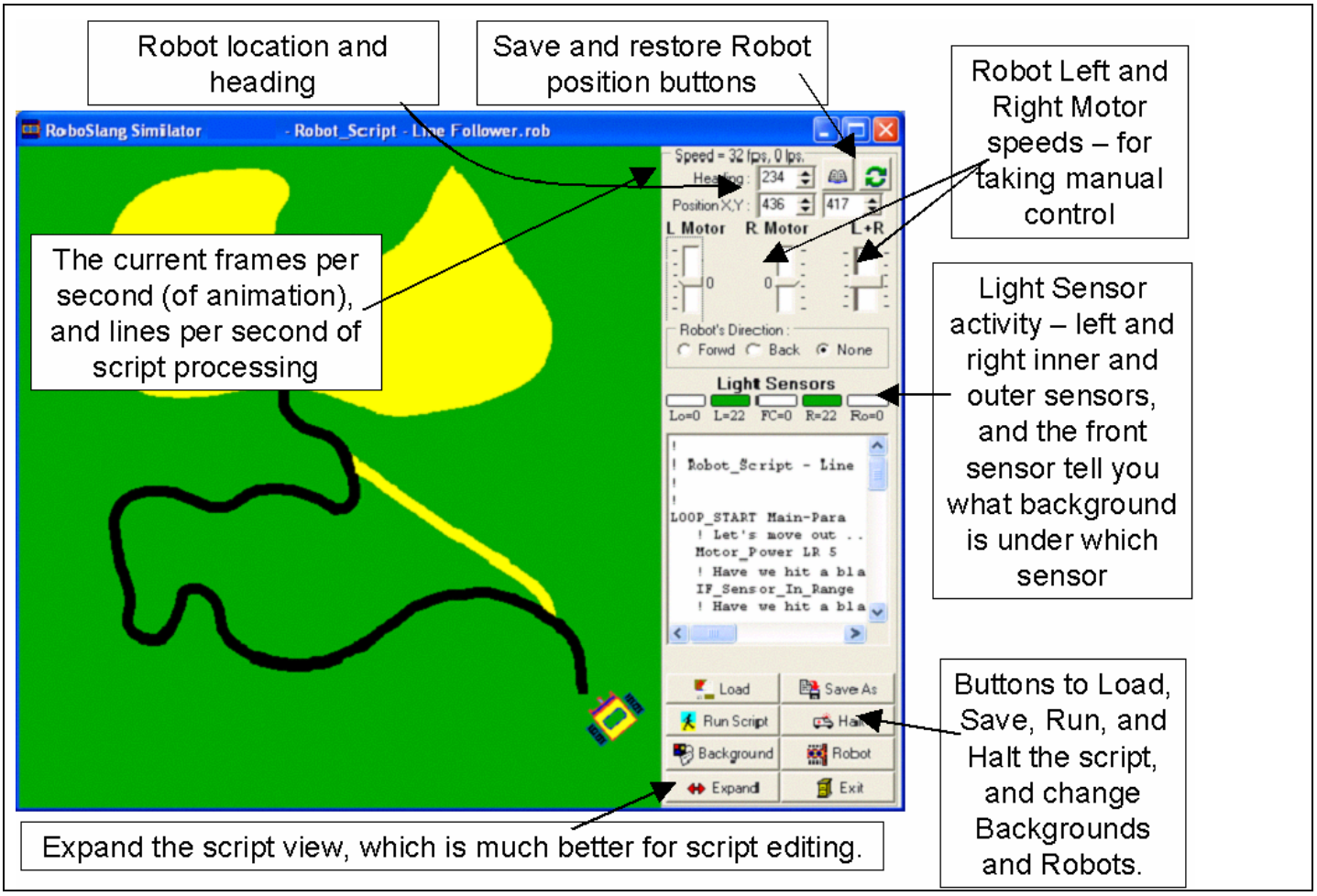
Here are some labelled screen shots which will help you understand how to work this program :

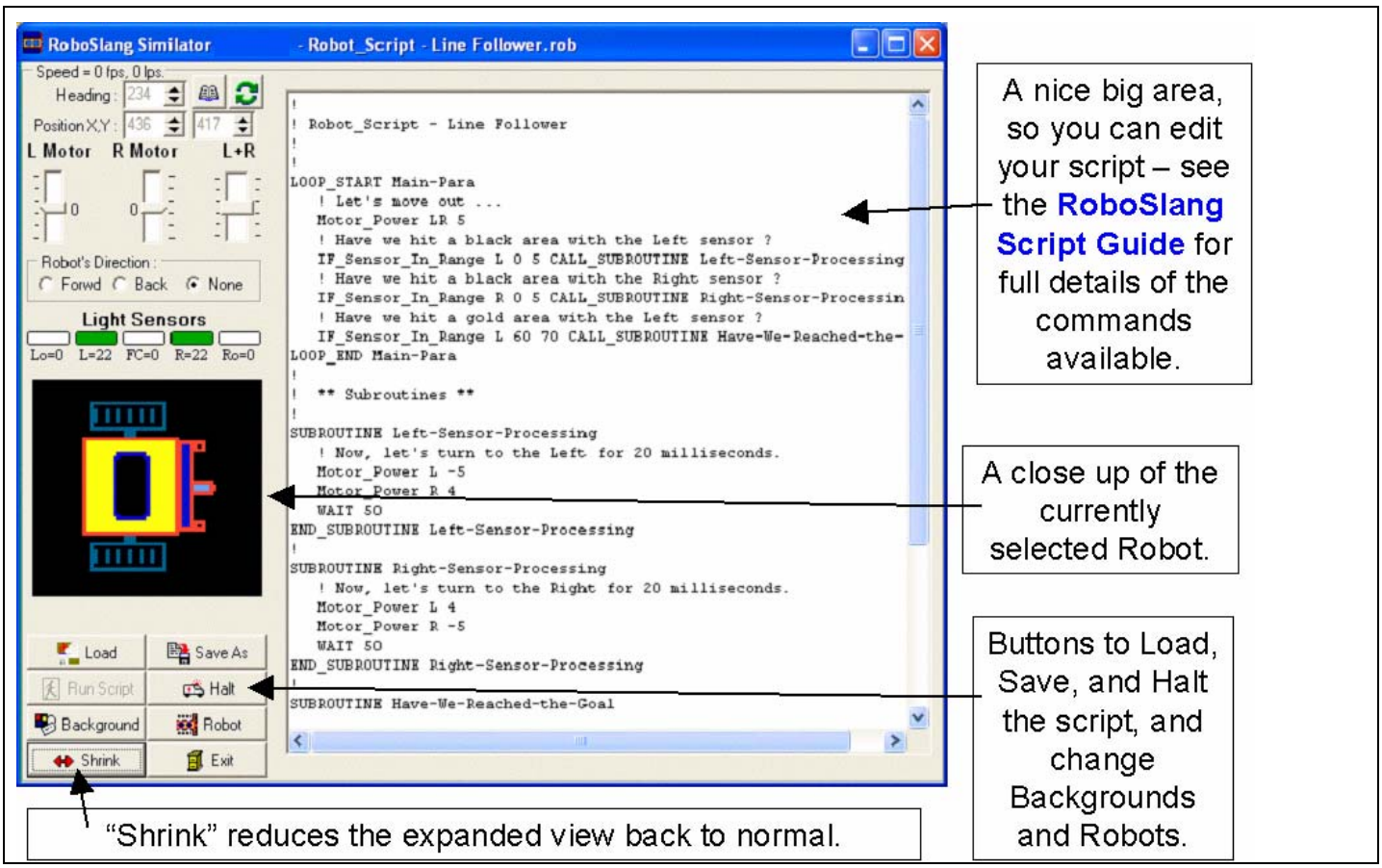
You can also control the robot manually - using the controls on the top right corner of the screen - by setting the angle and moving the sliders to change the speeds of the motors on each side of the robot.

Use the "Expand" button to get a much bigger view of your script – makes editing much easier.

When expanded, a "Shrink" button will appear, allowing you to return the view to normal.

See the "RoboSlang – Introduction" below and the "RoboSlang_Script_Command_Guide.pdf" for further details of the RoboSlang language and what the Light Sensor positions mean.



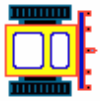


Available Robots

N.B. See the "RoboSlang – Introduction" below and the "RoboSlang_Script_Command_Guide.pdf" for further details of the RoboSlang language and what the Light Sensor positions mean.

Also, note that this document might be out of date, and that additional Robots may have been added since this was written – see my web site for details.

	Nickname: Filename: Number of Light Sensors: Light Sensors Positions:	**Square Bear** Robot - Square Bear.bmr 2 L, R
	Nickname: Filename: Number of Light Sensors: Light Sensors Positions:	**Skinny Minnie** Robot - Skinny Minnie.bmr 2 L, R
	Nickname: Filename: Number of Light Sensors: Light Sensors Positions:	**Hunch Back** Robot - Hunch Back.bmr 2 L, R

	Nickname:	**Big Bruiser**
	Filename:	Robot - Big Bruiser.bmr
	Number of Light Sensors:	5
	Light Sensors Positions:	Lo, L, FC, R, Ro
	Nickname:	**Moose Mobile**
	Filename:	Robot - Moose Mobile.bmr
	Number of Light Sensors:	5
	Light Sensors Positions:	Lo, L, FC, R, Ro

RoboSlang - Introduction

RoboSlang is an easy to understand language which can be used to control your virtual robot.

Why is my script language called "RoboSlang" ? Well, I just thought that it was time that our synthetic friends started gettin' down and talkin' jive baby ! ;)

There are a bunch of useful RoboSlang commands that you can use in such a script - more are being added.

So, for example, to make the Robot spin as fast as possible clockwise for 2 seconds, you would make the left motor go fast forward and the right motor go fast backwards, as follows in RoboSlang :

```
Motor_Power L 10
Motor_Power R -10
WAIT 2000
```

To make the robot move in a nice graceful arc for 5.25 seconds, you could do this in RoboSlang :

```
Motor_Power L 5
Motor_Power R 2
WAIT 5250
```

To get the robot moving in a nice straight-line, you could do this in RoboSlang :

```
Motor_Power L 3
Motor_Power R 3
```

Or, in a single RoboSlang step :

```
Motor_Power LR 3
```

To stop Motor R (the motor on the right side of the robot), you could this in RoboSlang :

```
Motor_STOP R
```

or this :

```
Motor_Power R 0
```

To stop both motors, you could do this :

```
Motor_STOP LR
```

or this :

```
Motor_Power LR 0
```

To turn the Robot slowly clockwise by (say) 45 degrees, you could do this in RoboSlang :

```
Turn_Right 45
```

Similarly, a to turn the Robot slowly anti-clockwise by (say) 130 degrees, you could do this in RoboSlang :

```
Turn_Left 130
```

To make the Robot move in an arc, stop, move in an arc, stop, 3 times, you could do this in RoboSlang :

```
Motor_Power L 5
Motor_Power R 1
WAIT 1000
Motor_Power L 5
Motor_Power R 1
WAIT 1000
Motor_Power L 5
Motor_Power R 1
WAIT 1000
```

Or, you could do this much more compactly in a Loop in RoboSlang :

```
LOOP_START Move_in_Arc 3
  Motor_Power L 5
  Motor_Power R 1
  WAIT 1000
LOOP_END Move_in_Arc
```

You can have loops within loops within loops. i.e. nested loops.

To make the Robot teleport to a new random location, you can do this in RoboSlang :

```
Teleport_Random_H
```

to move to Robot to a new, random horizontal location, or :

```
Teleport_Random_V
```

to move to Robot to a new, random vertical location, or :

```
Teleport_Random_H
Teleport_Random_V
```

to move to Robot to a new, random horizontal and location, or :

```
Teleport_Random_HV
```

to achieve this in a single RoboSlang step.

You can also do conditional testing of the inputs from the light sensors (which are at the front corners of the robot and are pointing at the ground so that they pick up the colour / brightness of the surface they are travelling over).

```
IF_Sensor_In_Range L 0 5 CALL_SUBROUTINE Left-Sensor-Processing
```

This statement means, if Sensor L (i.e. the left sensor) is over a very dark colour (very close to black), then perform the processing in the Left-Sensor-Processing subroutine.

This subroutine would then be contained within these statements :

```
SUBROUTINE Left-Sensor-Processing
  ::: RoboSlang statements
RETURN_FROM Left-Sensor-Processing
```

After the subroutine has run (i.e. when the RETURN_FROM statement is hit, then control is passed back to the line of code AFTER the IF_Sensor_In_Range statement.

Finally, you can also halt both the script and the robot immediately in RoboSlang with :

```
HALT
```

There is a limit of one command / stament per line in RoboSlang. So, for example all of these commands on one line would be invalid :

```
Motor_Power L 1   Motor_Power R 5  Turn_Left 20
```

and only the first command would be executed on this line, and the others would be flagged as errors.

At the end of the RoboSlang script, the robot will continue moving along unless you tell the motors to stop (at the end of the RoboSlang script).

Included with the program are a bunch of sample RoboSlang scripts - which demonstrate these commands.

Study these sample RoboSlang scripts and pretty soon, you will be a master RoboSlang script writer ! :)

You can change the position of the Robot at any time by simply clicking the left-mouse button anywhere in the main program window.

You can also Halt, Run, Save, and Load scripts.

Oh, yes, one last thing, none of the commands are case sensitive. Case sensitive programming languages suck !! So these commands have precisely the same effect :

```
Motor_Power R 1
MOTOR_pOwEr r 1
motor_power r 1
MOTOR_POWER R 1
```

And, the language is very flexible and forgiving (unlike the vast majority of programming languages), and any junk (){}<>[].,;:), and any spaces or tabs at the start and end of each line, and multiple spaces within a line are automatically removed. So these commands have precisely the same effect :

```
Motor_Power R 1
Motor_Power   < r >    {1}
[<Motor_Power> {([.r, ;  1  :  } ]
```

How's that for flexibility !! ;)

Also, any lines starting with the characters "!", "#", or "$", will be treated as comments and ignored. So, for example, these lines are all treated as comments :

```
! Start the Robot moving forward ...
# What is under the left sensor ?
$ Reverse and turn ...
```

Included with the program is the "RoboSlang_Script_Command_Guide.pdf" which contains a full list of the available commands, along with their meanings, examples, etc.

See the "**RoboSlang_Script_Command_Guide.pdf**" for further details of the RoboSlang language.

Calibrating your Robot

The IF... command above can be used to determine the colour of the background under the Robot.

Depending on the background image you have chosen, you may need to calibrate the sensors to work with the image.

To do this, left-click the mouse at various places on the background to move the robot around, and keep an eye on the left / right light sensor values. You can then alter the IF... tests in your script to use the required values to match this background.

Future Changes / Still to Do / Work Outstanding

Many changes / improvements / additions could be made to this program in the future, such as :

1. More robots – with more sensors and other advanced options.
2. More robot **attachments and tools**, for example :
   - compass,
   - GPS,
   - temperature probe,
   - collision sensor,
   - flashlight,
   - etc.
3. The ability to design your own robots. i.e. a Robot Editor.
4. More backgrounds – e.g. a **Robot Soccer** Field.
5. Support for **multiple robots** on screen at once – each with their own script.
6. Support for Robots working together as teams. e.g. Robot Soccer.
7. Computer controlled robots. e.g. so you can pit your robots against the computer in Robot Soccer, or follow the line or whatever.
8. Ability to equip the Robots with **weapons** – armour, shields, lasers, guns, mines, missiles, etc.
9. **Power-ups** – e.g. speed ups, weapons upgrades, etc
10. **3D** – move to 3D terrain / environments and 3D robots.
11. Anything else ?

The sky really is the limit with this program, and I could add in some terrific features and all sorts of fun functionality.

If you would like any of these improvements, or would like to suggest more, please email me and let me know.

Implementing some of the above functionality will require many months of hard work.

How much more work I do on this program depends entirely on what support I get. i.e. how many people use the program, etc.

Reviews / Awards / CDs

- v0.0080 of this program was demonstrated and show cased on a big projection screen before current and prospective students, staff, and the general public all day during **Central Queensland University's Open Day**, in Rockhampton, Australia, on August 3, 2004.

Freeware Information

This is free software.

Anyone - any person, any company, or any business - can use this program for free. No fees or payment is required.

However, if you find the program useful, then please consider making a PayPal donation to support my efforts. (To make a donation, please run the program and select the "About" option under the Help menu, and then click the PayPal link on the "About" screen).

Warranty

This software and the accompanying files are provided "as is" and without warranties as to performance or merchantability or any other warranties whether expressed or implied. The user assumes the entire risk of using this software.

If you do find any faults with this program, email me and let me know.

Amendment History

Version	Release Date	Changes / Comments
v1.0	9-May-2005	First Public Release. (67,749 lines of code / comments.)
V1.0f	18-Sep-2007	This program is now FREEWARE - see "Freeware Information" above.

If this program was not downloaded from my Home Page, then it is possibly an old version. In addition, there may be additional features and robots in a newer version. The latest version of this program – with the latest features and robots - is available from my WEB page - see below.

If you enjoy using this program, and/or would like to support the continued development of useful software, please consider making a small donation via **PayPal** (http://www.paypal.com/) to moose@move.to

All the best,

Mike "Moose" O'Malley

Moose's Software Valley - Established July, 1996.
WEB: http://move.to/moose