NodesWin Version 1.0 User Guide

Contents

1. Introduction

NodesWin is a windows version of the Nodes simulation software. The paper 'Nodes: An Environment for Simulating Kinematic Self-Replicating Machines' contains an introduction to the Nodes environment.

NodesWin allows you to open, run and save simulations. You can navigate around the universe by clicking the mouse at the point in the universe that you want to move to the centre of the active window. You can also zoom in and zoom out.

2. Menu Commands

File
  Open...
    Prompts for a `.inp` or `.dmp` file.

  Close

  Save as...
    Saves the state of the universe in the active window as a `.dmp` file.

  Exit

Edit
  Apply changes...
    Prompts you for an `.rpl` file. Changes specified in this file will be applied to the universe in the active window.

View
  Zoom in...

Zooms in on the centre of the active window.

Zoom out...
Zooms out from the centre of the active window.

Show signals
Show the signals at terminals of nodes. Terminals producing no signal will be shown as white discs. Terminals producing signals will be shown as red discs.

Show signal values
Show the values of non-zero signals at terminals of nodes.

Run
Go
Start running the simulation.

Stop
Stop running the simulation.

Step
Run a single simulation iteration step.

Tools
Animation
Allows you to specify a filename for an animation output file, and specify how frequently a new frame is added to the animation file. An animation file frame consists of a line of three integers for every node in the universe. These three integers specify the type, x-coordinate and y-coordintate of a node. Frames are delimited with a line containing a single value of -1.

3. Input file formats

Two types of file can be used as inputs to nodes.

3.1 `.inp` input files

`.inp` input files use a simple file format and can easily be created using a text editor or by an automatic script.

`.inp` input files contain any number of lines, each containing one of the following commands:

`node <identifier> <type> <x> <y> <orientation> <radius>`

Create a node of `<type>` at position `<x>`,`<y>` with the specified `<orientation>` (in radians) and `<radius>`. The `<identifier>` will

be used to refer to this node later on in the `.inp` file. See appendix A for a list of node types. Nodes has never been used with any `<radius>` other than 8, so using a different value may not work.

`nodeatpos <identifier> <x> <y>`

From this point forwards, `<identifier>` will be used to refer to the node at position `<x>`,`<y>`

`connect <identifier1> <identifier2>`

Connect together the two specified nodes. The terminals used for the connection will be the two which are most closely aligned.

`setnum <identifier> <value>`

Set the internal state of the specified node to the specified value. Most nodes do not have an internal state, this command will only affect those that do. ('toggle' and 'store' nodes).

3.2 `.dmp` input files and `.rpl` files

The state of a universe may be written to a `.dmp` file so that nodes can be restarted from the same state later on.

An `.rpl` file may be used to replace specific nodes in an `.dmp` file (or a `.inp` file), or to remove nodes from specified areas of the universe. For example, you may have produced a dump file at a particular point in a simulation, and you may want to start the simulation from that point, but alter a few nodes or remove nodes from a part of the simulation that you are not interested in. An `.rpl` file is used for this purpose.
An `.rpl` file contains any number of lines of the forms:

`exc <x1> <y1> <x2> <y2>`

Remove all nodes in the region defined by `x1`,`y1`,`x2`,`y2` where `x1<x2` and `y1<y2`

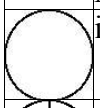
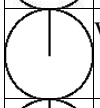
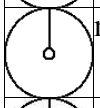
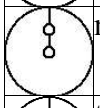
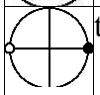
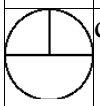
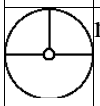
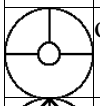
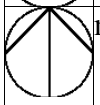
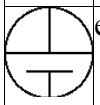
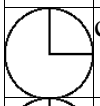
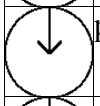
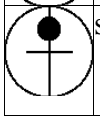
`rep <x> <y> <type> <orientation>`

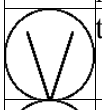
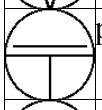
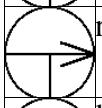
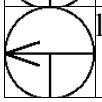
Replace a node with a node of a different type, where `<x>` and `<y>` are the coordinates of the node to replace, `<type>` is the type of the replacement node (see appendix A for a list of types). And `<orientation>` is the orientation of the replacement node (a number in the range 0-3). The replacement node will obviously have its terminals in the same positions as the node that it is replacing, but within this constraint, 0 means orient the node so that its north terminal points as nearly as possible in the positive y direction, 1 means orient it 90 degrees clockwise from this and so on for 2 and 3.

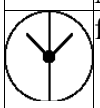
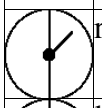
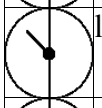
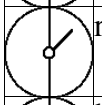
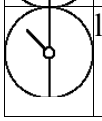
The universe coordinates of the mouse pointer appear at the right of the status bar.

Appendix A. Node Types

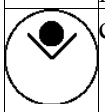
The following node types are implemented in Nodes Version 1.0:

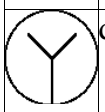
Wiring and Logic			
Node	**Short Name**	**Long Name**	**Function**
	ins	insulator	none
	wire	wire	output[north] = input[south]
	not	inverter	output[north] = !input[south]
	notnot	double-inverter	output[north] = !!input[south]
	toggle	toggle	if input[east], set output[north] and output[south] to 1 if input[west] set output[north] and output[south] to 0
	delta	delta	output[north] = input[south] output[east] = input[south] output[west] = input[south]
	ndelta	delta-inverter	output[north] = !input[south] output[east] = !input[south] output[west] = !input[south]
	or	adder	output[north] = input[east]+input[west]
	maj	majority	output[north] = input[east] * input[west] + input[east] * input[south] + input[west] * input[south]
	equal	comparator	output[north] = ((input[east] == input[west]) && input[east] != 0) || ((input[east] == input[south]) && input[east] != 0) || ((input[south] == input[west]) && input[south] != 0)
	cross	crossover	output[north] = input[south] output[east] = input[west]
	pulse	pulser	if a rising edge is detected on input[south], output[north] = 1, otherwise output[north] = 0
	store	store	if output[north] is 0 and input[south] is non-zero then set output[north] to input[south]. if input[east] && input[west] then set output[north] to 0

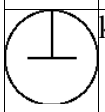
Forces			
Node	**Short Name**	**Long Name**	**Function**
	thrust	thruster	if (input[south]) apply force on self in southward direction
	push	pusher	if (input[south]) then apply northward-directed force on the node to the north (if any)
	rslide	right-slider	if (input[south]) then apply eastward-directed force on the node to the north (if any)
	lslide	left-slider	if (input[south]) then apply westward-directed force on the node to the north (if any)

Connections			
Node	**Short Name**	**Long Name**	**Function**
	fuse	fuser	if there are nodes to the north and north-east, connect them together when input[south] is non-zero also, if there are nodes to the north and north-west, connect them together when input[south] is non-zero
	rfuse	right-fuser	if there are nodes to the north and north-east, connect them together when input[south] is non-zero
	lfuse	left-fuser	if there are nodes to the north and north-west, connect them together when input[south] is non-zero
	runfuse	right-unfuser	if there are nodes to the north and north-east, disconnect them when input[south] is non-zero
	lunfuse	left-unfuser	if there are nodes to the north and north-west, disconnect them when input[south] is non-zero

Others			
Node	**Short Name**	**Long Name**	**Function**

	creator	creator	if the value of input[south] is not zero, and there is no node within one node diameter of the point one node diameter north of the creator node, create a node at this point.
			
			The type and orientation of the newly created node are determined by the value of input[south]. See appendix B for details.
			
			`Type = (input[south]-1) / 4`
			`Orientation = orientation of creator node +`
			`((input[south]-1) % 4) * 90 degrees`

	detect	detector	if there is a node within one node diameter of the point one node diameter north of the detector node, output a signal determined by the nodes type and orientation, otherwise output zero.
			
			`output[south] = 1 + type * 4 +`
			`orientation / 90 degrees`
			
			For the purpose of this calculation, the orientation is rounded to the nearest ninety degrees.

	keypress	keypress	if a keyboard is connected to the computer, and the numeric key matching the value of input[south] is pressed, output[north] = 1, otherwise output[north] = 0.

Appendix B. Node Codes

The following encoding is used by the creator and detector nodes.

```
0 = store       1 = ins         2 = wire        3 = not
4 = thrust      5 = fuse        6 = rfuse       7 = lfuse
8 = toggle      9 = delta       10 = ndelta     11 = detect
12 = or         13 = runfuse    14 = lunfuse    15 = cross
16 = keypress   17 = pulse      18 = rslide     19 = lslide
20 = creator    21 = maj        22 = notnot     23 = push
24 = equal
```