RoboSim User's Guide

Version 1.7.70

Contents

1 Introduction

`RoboSim` is a robot simulation environment, developed by the UC Davis Center for Integrated Computing and STEM Education (C-STEM) **(http://c-stem.ucdavis.edu)**, for programming Barobo Mobot and Linkbot. The same Ch program can control hardware robots or virtual robots in RoboSim without any modification.

2 RoboSim GUI

RoboSim can be conveniently launched by double clicking its icon RS on the desktop. The RoboSim graphical user interface (GUI), shown in Figure 1, allows the user to change between hardware and virtual robots when a Ch robot program is executed. There is no save button within the GUI, all changes made are automatically saved.

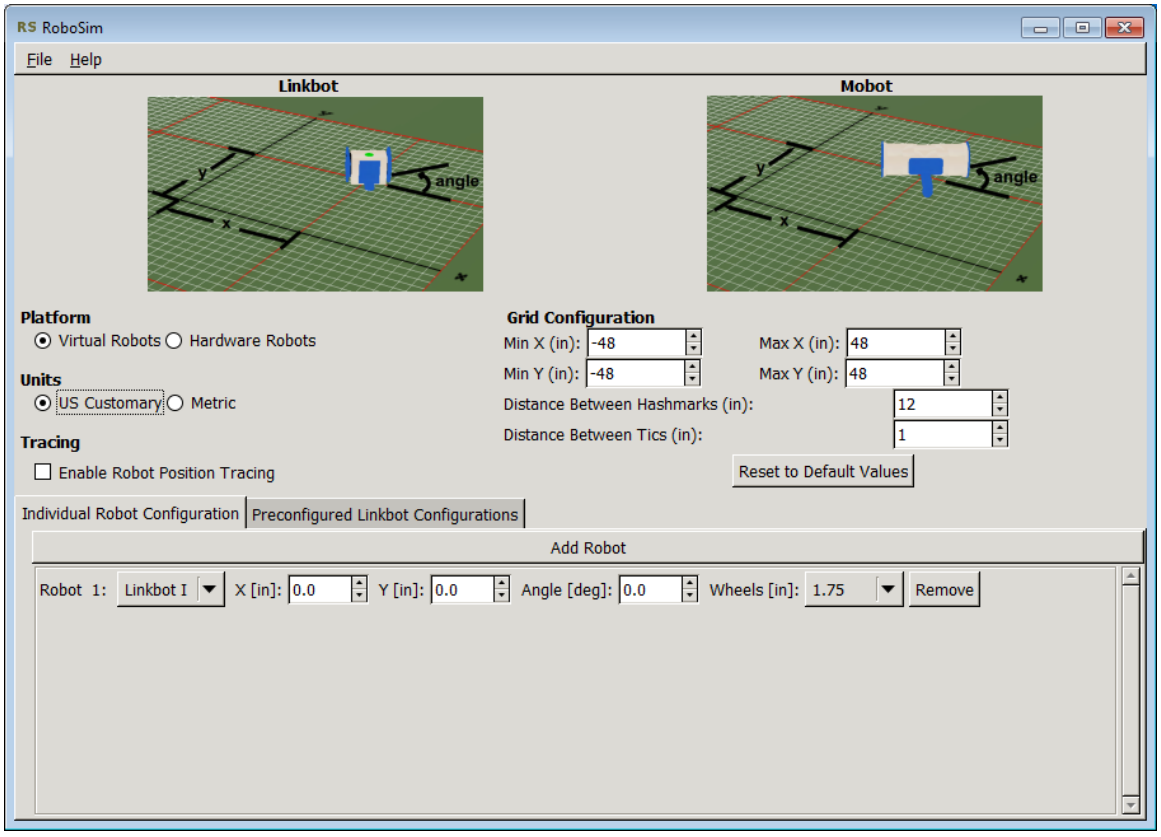

Figure 1: The RoboSim GUI.

2.1 Platform

The **Platform** entry as shown in Figure 2, allows the user to decide whether a Ch program controls the hardware or virtual robots. Each time a new Ch program is started, it will check the setup based on this entry. For a Ch robot program to control a virtual robot, check the box for **Simulated Robots**. If the box for **Hardware Robots** is checked, a Ch program will control the physical hardware robots.

Figure 2: Initial robot configuration dialog.

2.2 Units

Simulations within RoboSim can be run either in **US Customary** units consisting of inches, degrees, and seconds or **Metric** units with centimeters, degrees, and seconds. Changing units will effect the grid spacing drawn beneath the robots and the spacing between robots. Changing between these two options will change the labels within the GUI to indicate the units being used.

2.3 Tracing

Tracing where robots have been can be enabled by selecting the check box 'Enable Robot Position Tracing', as shown in Figure 1. When the tracing is enabled, lines following the robot trajectories will be drawn for each robot. Mobot tracking lines will be in a green color and Linkbot tracking lines will be in the color matching the Linkbot LED.

2.4 Grid Configuration

To be able to see how far robots have moved, a grid is enabled under the robots. There are six options to alter the layout of the grid lines under the **Grid Configuration**. The minimum and maximum extends of the grid for both the X and Y directions can be specified individually. Rectangular grids of any size can be created in any of the quadrants. Hashmarks are the red lines drawn within the configuration images. By default, the distance between two hashmarks is 12 inches in US Customary units and 50 centimeters in Metric units. Tics are the most frequent lines drawn in a light gray. By default, the distance between two tics is 1 inch in US Customary units and 5 centimeters in Metric units.

Switching between US Customary and Metric units will change these default values to logical starting points for the metric system. The 'Reset to Defaults' button will allow the default values for both US Customary and Metric to be reinstated after they have been changed. Depending upon which units are currently selected from Section 2.2, either the US Customary defaults, shown in Figure 3, or the Metric defaults, as shown in Figure 4, will be set.

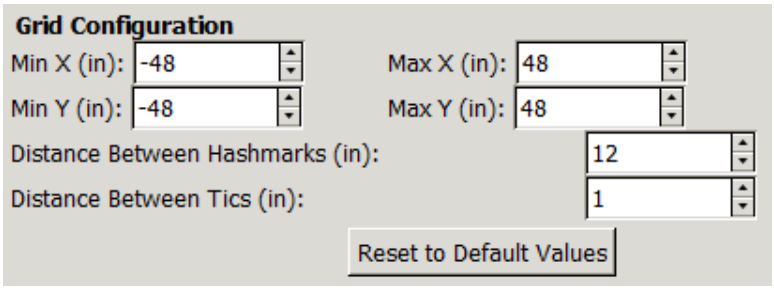

Figure 3: Default US Customary Grid Spacing.

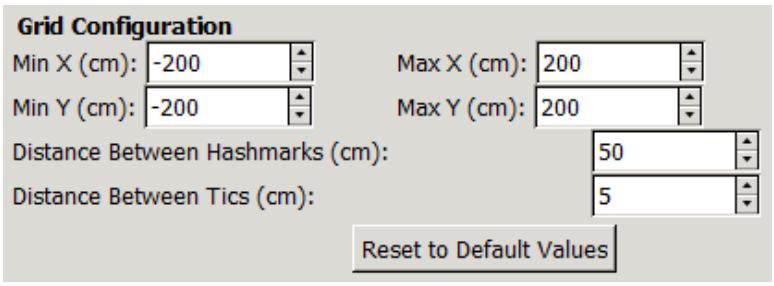

Figure 4: Default Metric Grid Spacing.

2.5 Individual Robot Configuration

Initial robot configurations can either be done through the **Individual Robot Configuration** or **Preconfigured Robot Configuration** tabs. The **Individual Robot Configuration** section, as shown in Figure 5, has options to allow robots to be positioned within the RoboSim scene either with or without wheels but not attached to each other.

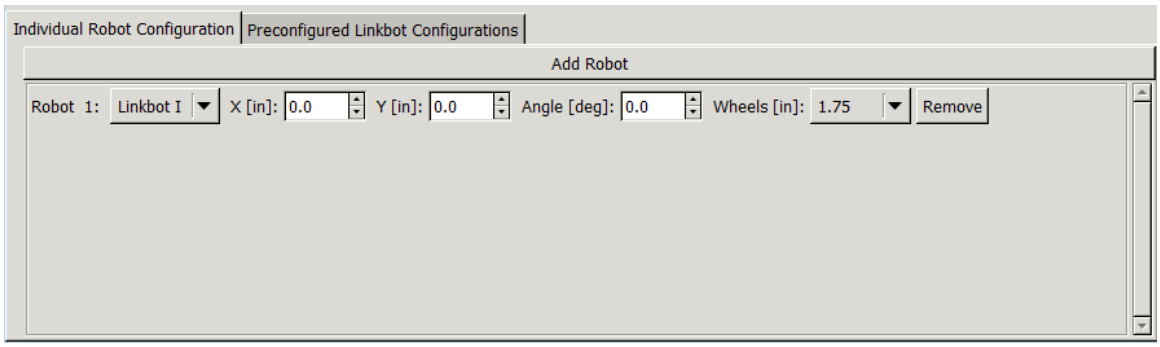

Figure 5: Individual robot configuration dialog.

The user can specify the X and Y coordinates as well as the orientation angle of a virtual robot. Images for the Linkbot and Mobot showing the meaning of each of the options are displayed above the configuration box. They are screenshots of the virtual robots positioned at one foot in both the X and Y coordinates with the orientation angle of 30 degrees from the X-axis.

Initially, the individual robot list contains one Linkbot-I at (0, 0) with 1.75 inch wheels. More robots can be added by the 'Add Robot' button. Clicking this button will add a robot into RoboSim, each offset from the previous one in the x-direction by 6 inches or 15 centimeters depending upon the units selected. The order within the robot list will be the order in which the robots will be read into the simulation program.

2.5.1 Robot Type

There are three options for robot type available. Linkbot-I, Linkbot-L, and Mobot. The options are presented in a drop down menu.

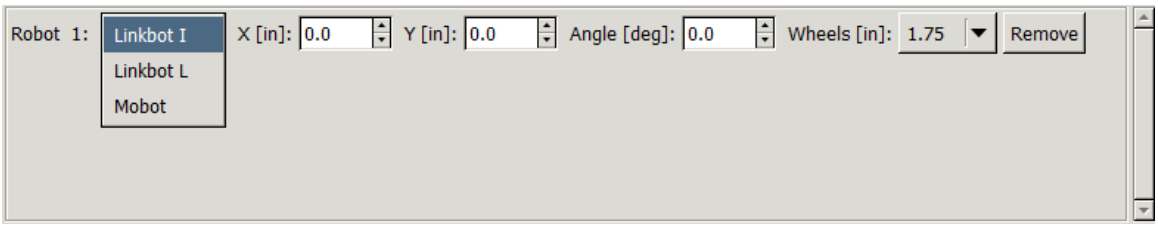

Figure 6: Picking a robot type.

2.5.2 Robot Position

Both X and Y positions can be chosen independently for each robot.

2.5.3 Robot Angle

The rotation angle from the x-axis can be used for changing the orientation of the two robots respective to each other.

2.5.4 Wheels

Since so many times the robots are run with wheels and a caster connected, a drop down menu is provided to select different wheel sizes. The options listed are the radii of the wheels provided with Linkbots when purchased from Barobo. Each wheel is drawn with a series of dots along the one radius to easily show the rotation of the wheel. The correlation between wheel radius and number of dots is given in Table 1.

Number of Dots	Wheel Radius
2	custom radius
3	1.625 inch / 4.13 centimeter
4	1.75 inch / 4.45 centimeter
5	2.00 inch / 5.08 centimeter

Table 1: Wheel sizes and number of dots.

Custom wheel sizes are available by using the 'Custom' option from the drop down menu. This option creates an input box to the right to let the user enter a wheel radius.

2.5.5 Remove

A robot can be removed from the RoboSim by clicking the 'Remove' button.

2.6 Preconfigured Robot Configurations

In addition to positioning robots independently within the RoboSim, some **Preconfigured Robot Configurations**, as shown in Figure 7, which represent commonly used Linkbot configurations are available to the user. Selecting one of these options will display a picture of the configuration built with the hardware Linkbots and corresponding to a Ch robot program presented in Chapter 13 in the book *Learning Robot Programming with Linkbot for the Absolute Beginner*. When one of these options is selected, the specific configuration for this setup is passed into Ch and robots specified in the individual robot configuration are ignored. To switch back to the individual configuration, just unselect the selected preconfigured robot configuration.

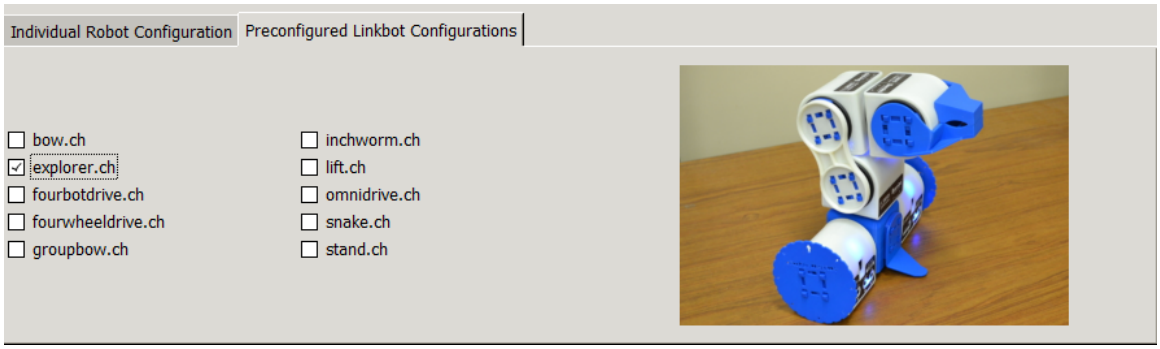

Figure 7: Preconfigured Linkbots.

3 Running a Ch Program with RoboSim

Once the simulation environment has been configured with the RoboSim GUI in Section 2, the user can run Ch programs in ChIDE to control the virtual robots. The RoboSim GUI should remain open while simulating robots. Once it is closed, the system will revert to hardware mode. The RoboSim scene with virtual robots for each simulation are created upon running a Ch program. For example, when the Ch program `moveforward3.ch` below

```
/* File: moveforward3.ch
   Move forward for Linkbot-I as a two-wheel vehicle */
#include <linkbot.h>
CLinkbotI robot;

/* connect to the paired robot and move to the zero position */
robot.connect();
robot.resetToZero();

/* move forward by rolling two wheels for 360 degrees */
robot.moveForward(360);
```

is executed in ChIDE, a RoboSim scene shown in Figure 8 will be displayed.

```
Paused: Press any key to start
```

is displayed in the RoboSim scene to reminder the user that the virtual robot will not move until the user presses any key on the keyboard. This gives the user an opportunity to examine the RoboSim scene before the motion begins.

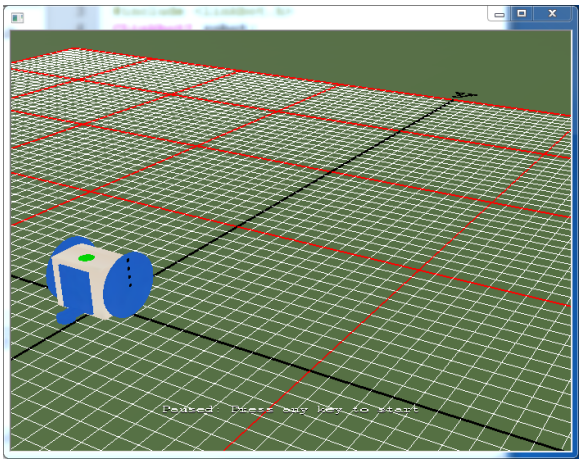

Figure 8: A RoboSim scene with a virtual robot at its starting position.

While a robot is moving in the RoboSim scene, the user can press any key to pause the motion of the robot. When the motion is paused, the message

```
Paused: Press any key to restart
```

will be displayed in the RoboSim scene. The user can press any key to restart the motion.

When the user presses the 't' key, the robot trajectory is tracked in a green line in the RoboSim scene as shown in Figure 9.

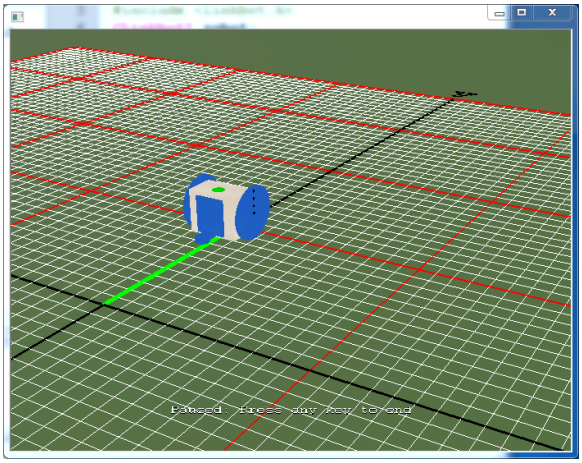

Figure 9: A RoboSim scene with a virtual robot and its trajectory tracked.

When the program is finished, the message

```
Paused: Press any key to end
```

will be displayed in the RoboSim scene. Pressing any key, the RoboSim scene will disappear.

4 Interacting with a RoboSim Scene

The user can interact with a RoboSim scene through the keyboard and mouse.

The ground plane is for reference only. It is designed to disappear when viewing the robots from below to be able to inspect the movement from all angles.

4.1 Keyboard Input

The RoboSim scene responds to keyboard input as outlined in Table 2. As described in the previous sections, the 't' key will toggle the tracking of robot trajectories.

key	action
1	return to home camera position
2	set camera to overhead view
n	toggle grid line numbering
r	toggle robot visibility and enable tracking
t	toggle robot tracking
any other key	Pause and unpause simulation

Table 2: Keyboard input for RoboSim

There are two views available to the user. The default view, which can be toggled with the '1' key, is from behind the robots looking into the first quadrant. This view can be seen in any of the RoboSim scene screenshots within this document, except for Figure 10 which shows the overhead view. The '2' key moves the camera directly above the origin looking down on the scene creating a 2D viewpoint of the robots.

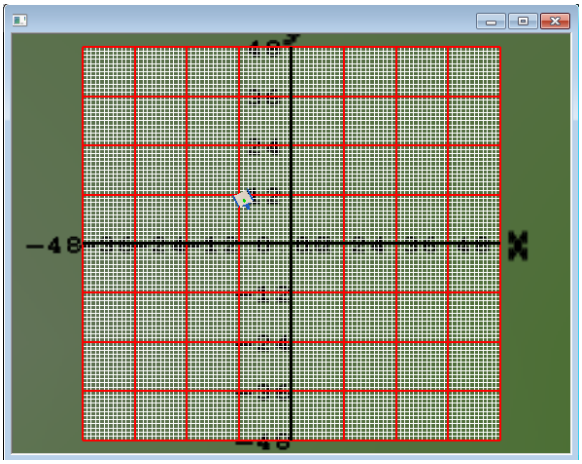

Figure 10: A RoboSim scene with the overhead viewing angle.

The 'n' key allows the user to toggle the display of the grid numbering. X and Y numbering is by default enabled and given for every hashmark on the grid.

The 'r' key will toggle the display of virtual robots or robot trajectories. It will also disable collision checking between robots so that trajectories can be intersecting. This feature is useful when the user would like to view a trajectory traced by a robot without the virtual root blocking the trajectory. Figure 11 shows a RoboSim scene with a tracked robot trajectory only.

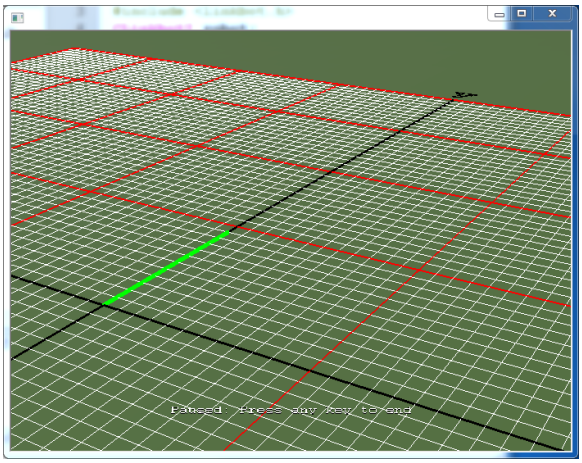

Figure 11: A RoboSim scene with a tracked robot trajectory only.

As described in the previous section, the motion of robots in the RoboSim scene can be paused and restarted by pressing any other key on the keyboard.

4.2 Mouse Input

Clicking on a robot in a RoboSim scene will enable a pop up which displays the robot number and the current position of the robot, as shown in Figure 12 with the position (0, 10.9817) inches for the X and Y coordinates for the Robot 1. Clicking again the displayed position for the robot will disappear.

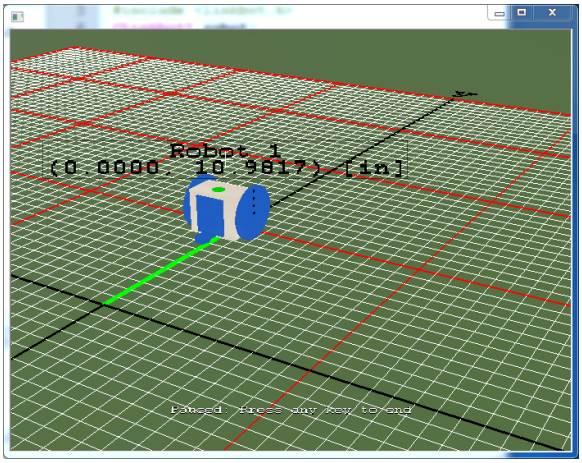

Figure 12: A RoboSim scene with a virtual robot and its position displayed.

The user can execute a Ch robot program in debug mode in ChIDE, line by line, with the command `Next`. At the end of each motion statement, the user can click the robot in the RoboSim scene to obtain the X and Y coordinates of the robot. The ability to obtain the X and Y coordinates of a robot during its motion along a trajectory can be very useful for learning many math concepts.

The mouse can be used to move the camera around the scene. Holding the left mouse button and dragging the mouse pans the camera as outlined in Table 3. Holding the right mouse button and dragging the mouse

enables scaling of the view by zooming in and out. Holding both left and right mouse buttons and dragging changes the location of the camera within the scene.

The ground plane is for reference only. The ground plane will disappear when viewing the robots from below so that the user can inspect the movement from all angles.

button	**action**
Hold left mouse button and drag	rotate camera
Hold right mouse button and drag	zoom in and out
Hold both left and right buttons, and drag	pan around scene
Click on a robot	display the robot position

Table 3: Mouse input for the RoboSim scene.

5 Differences to Hardware

RoboSim has some different options which affect the performance of the simulation. These are options which are passed into the connect function of the robots that are not available within hardware. The RoboSim GUI writes all of the options into a user configuration file that has to be changed each time a new code is to be run. However, users can manually write configuration files specific to each code. To tell RoboSim to use a custom configuration file over the global one, pass the name of the file into the first connect function call in the file. RoboSim will look in the same folder to find the configuration file, if it is not found, then it will use the default one.

```
#include <linkbot.h>
CLinkbotI robot;

robot.connect("testrc");
```

A Manual Configuration File Generation

All options available in the RoboSim GUI can be done manually as well as adding in more advanced features which are not necessary to be exposed to all users in the GUI. There are four sections to the configuration file. The header, configuration, ground, and simulation sections. Each holds specific types of information about the simulation to be run.

A.1 Header

The header is one line and specifies to RoboSim that this is a valid xml file. This line should be placed at the start of each simulation file.

```
<?xml version="1.0" encoding="UTF-8"?>
```

A.2 Configuration Section

General parameters about the simulation can be added within the config section. Each one is its own line placed between the starting `<config>` and ending `</config>`.

```
<config>
</config>
```

A.2.1 Version

```
<version val="1"/>
```

The version of the XML configuration file. Updated internally when new non-backwards compatible changes are made.

A.2.2 Pause

```
<pause val="1"/>
```

The simulation can either start paused waiting for a user to press a key to start or it will start as soon as it is ready. Passing 0 to this option starts the simulation; while 1 pauses it at the beginning.

A.2.3 Real Time

```
<realtime val="1"/>
```

The simulation can run faster than real time. Passing 0 to this option allows the simulation to run as fast as the computer can simulate it.

A.2.4 Mu

```
<mu ground="0.9" body="0.3"/>
```

The coefficient of friction can be altered between the robots and the ground and between robots themselves. The default values are given here.

A.2.5 Coefficient of Restitution

```
<cor ground="0.3" body="0.3"/>
```

The coefficient of restitution can be altered between the robots and the ground and between robots themselves. The default values are given here. The COR is a measure of how bouncy a surface is. Small values correspond to hard surfaces while larger values are for softer surfaces.

A.3 Ground Section

The ground section holds the solid bodies which are a part of the ground for which the robots to interact. The objects do not need to be on the ground plane. They can be floating in space but are still a part of the ground objects. Each object has a mass, given in kilograms, associated with it and interacts in the simulation. An object with heavy mass cannot be moved by the robots, while lighter objects will be pushed around. There is a special zero mass option which will disable the object from interacting with the gravity of the simulation allowing floating objects. The RGB values for the coloring are on a scale of [0-1] as is the alpha blending parameter. An alpha value of 1 is fully opaque while a value of 0 is fully transparent. Everything to be added is put between the `<ground>` tags.

```
<ground>
</ground>
```

A.3.1 Box

```
<box mass="1">
    <color r="1" g="1" b="1" alpha="1"/>
    <size x="1" y="1" z="1"/>
    <position x="1" y="1" z="1"/>
    <rotation psi="1" theta="1" phi="1"/>
</box>
```

The ground box is configured with the mass, color, size, position, and rotation parameters. Mass is given as the total mass for the object. The color parameter gives the RGB values in which the object is going to be drawn. The size gives the lengths in the X, Y, and Z directions. Position gives the X, Y, and Z location of the center of the box. The rotation gives the three Euler Angles of the box.

A.3.2 Cylinder

```
<cylinder mass="1" axis="1">
    <color r="1" g="1" b="1" alpha="1"/>
    <size radius="1" length="10"/>
    <position x="1" y="1" z="1"/>
    <rotation psi="1" theta="1" phi="1"/>
</cylinder>
```

The ground cylinder is configured with the mass, axis, color, size, position, and rotation parameters. Mass is the total mass of the object. Axis is the long axis for the object given the convention 1=X, 2=Y, and 3=Z. The color parameter gives the RGB values in which the object is going to be drawn. The size gives the radius and length of the cylinder. Position gives the X, Y, and Z location of the center of the cylinder. The rotation gives the three Euler Angles.

A.3.3 Sphere

```
<sphere mass="1">
    <color r="1" g="1" b="1" alpha="1"/>
    <size radius="1"/>
    <position x="1" y="1" z="1"/>
</sphere>
```

The ground sphere is configured with the mass, color, size, and position parameters. Mass is the total mass of the object. The color parameter gives the RGB values in which the object is going to be drawn. The size gives the radius of the sphere. By default the cylinder is drawn with the long axis along the X-axis. Position gives the X, Y, and Z location of the center of the sphere.

A.4 Graphics Section

The graphics section hold elements which are drawn onto the screen but which do not interact with the robot simulation. They are there merely for visual representation and elucidation of the mathematics happening.

A.4.1 Grid

```
<grid units="1" major="12" tics="1" minx="-48" maxx="48" miny="-48" maxy="48"/>
```

The grid boxes from the GUI put their information here. `units`: 1 for US Customary and 0 for Metric. `major` are the red hashmarks and `tics` are the gray tick marks. `minx` and `maxx` are the minimum and maximum distances along the X-axis for the grids, respectively. `miny` and `maxy` are the complements for the Y-axis.

A.4.2 Tracking

```
<tracking val="1"/>
```

Setting to track robot location with lines on the ground. 1 for on; 0 for off.

A.4.3 Line

```
<line width="1">
    <color r="1" g="1" b="1" alpha="1"/>
    <start x="1" y="1" z="1"/>
    <end x="1" y="1" z="1"/>
</line>
```

The line is drawn from a starting XYZ (`<start>`) to an ending XYZ (`<end>`). Line width is an integer which scales the thickness of the line. Its color and transparency is affected by the `<color>` element each value of which scales from zero to one.

A.4.4 Point

```
<point size="1">
    <color r="1" g="1" b="1" alpha="1"/>
    <position x="1" y="1" z="1"/>
</point>
```

The point is drawn from at a XYZ position (`<position>`). Size is an integer which scales the radius of the point. Its color and transparency is affected by the `<color>` element each value of which scales from zero to one.

A.4.5 Text

```
<hello>
    <color r="1" g="1" b="1" alpha="1"/>
    <position x="1" y="1" z="1"/>
</hello>
```

Any text can be drawn onto the screen through specific XML elements. The name of the element, such as `hello` in this case, is the text to be rendered. Changing this element name (both the starting and ending tags) will change the text to be displayed. The text is drawn at the XYZ position in the `<position>` element. Its color and transparency is affected by the `<color>` element each value of which scales from zero to one.

A.5 Simulation Section

The simulation section holds the robots and accessories for the current simulation. Everything to be added is put between the `<sim>` tags.

```
<sim>
</sim>
```

A.5.1 Type

```
<sim type="0"/>
```

There are two options: either preconfigured robots or individual robots. Used to load the right options when launching the GUI. Can be 0 to show that the robots within the section are individual. Any number above that represents the preconfigured robots within the GUI.

A.5.2 Robots

Each robot has its own xml tag to configure its position, orientation, and rotation in space. The types of robot tags are tabulated in 4. All robots much have attributes associated with them, and optionally positioning arguments. The Linkbot-T is a Linkbot with all three joints being able to actuate.

`<linkboti/>`
`<linkbotl/>`
`<linkbott/>`
`<mobot/>`

Table 4: Robots

Robot Attributes
Each robot element is required to have one attribute titled **id** which is an unique identifier for the simulation to reference. A second optional attribute is **orientation** which orients the face of a second robot when it is being attached to a first robot. A third optional argument is **ground** which specifies which body part of the robot is attached to the ground. A fixed, permanent joint is created between this body part and the ground.

`<linkboti id="0"/>`	one Linkbot I with id = 0
`<linkboti id="0" orientation="3"/>`	Linkbot I is 'upside-down'
`<linkboti id="0" ground="3"/>`	Linkbot I's face 3 is fixed

Table 5: Examples

attribute	values	description
id	unique integer	a unique integer to identify each robot
orientation	1	robot face number is at 12 o'clock
	2	robot face number is at 3 o'clock
	3	robot face number is at 6 o'clock
	4	robot face number is at 9 o'clock
ground	0	body is attached to ground
	1	face 1 is attached to ground
	2	face 2 is attached to ground
	3	face 3 is attached to ground

Table 6: Robot Attributes

Robot Positioning
In addition to IDing each robot, they can be positioned in space independently of each other. There are sub-tags which specify the global attributes of the robot. `<position>` specifies the X, Y, and Z coordinates of the robot. `<rotation>` specifies the three Euler Angles of the robot about the X (psi), Y (theta), and Z (phi) axes. `joint` allows the joints to be rotated initially as opposed to being built at zero rotation. Examples of Linkbot and Mobot are shown below.

```
<linkboti id="0">
    <position x="0" y="0" z="0"/>
    <rotation psi="0" theta="0" phi="0"/>
    <joint f1="-20" f2="0" f3="20"/>
</linkboti>

<mobot id="0">
    <position x="0" y="0" z="0"/>
    <rotation psi="0" theta="0" phi="0"/>
    <joint a1="-20" a2="0" a3="20" a4="45"/>
</mobot>
```

Attaching Robots
When a second robot is going to be attached to the first robot, the only required argument is the id number. The positioning will be figured out automatically. The code below shows the configuration for the inchworm linkbot configuration. The first robot is placed at the origin and the second one is positioned by the code since it is attached to the bridge connector to the first one.

```
<linkbotl id="0">
    <position x="0" y="0" z="0"/>
    <rotation psi="0" theta="180" phi="0"/>
</linkbotl>
<linkbotl id="1"/>
<bridge>
    <side id="1" robot="0" face="1"/>
    <side id="2" robot="1" face="1"/>
</bridge>
```

All robots have faces onto which accessories can connect. These are the points of reference on how the robot will be positioned in space when attaching to another robot. The Mobot has eight connection faces, as shown in Figure 13. Faces 3 and 6 span the two body joints and thus provide a physical connection which

prevents the Mobot from bending its body. The three Linkbot faces, shown in Figure 14, are on each of the three sides of the robot. Two will be attached to the rotating faces of the robot depending upon whether the robot is a Linkbot-I or a Linkbot-L.

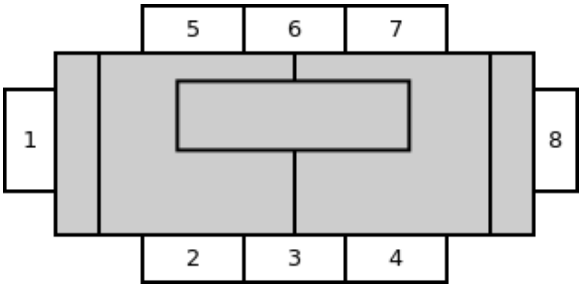

Figure 13: Mobot Connection Locations.

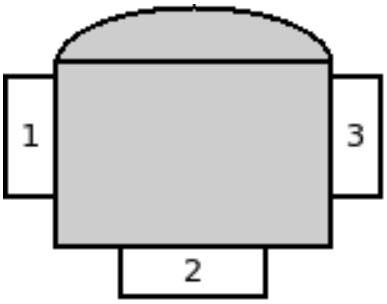

Figure 14: Linkbot Connection Locations.

A.5.3 Accessories

The accessories of the robots can be added to the scene and attached to the robots by the tags. Most accessories have multiple sides which are each configured independently. The options for a side are its id, robot which is attached to that side, and the face of that robot.

```
<side id="1" robot="1" face="1"/>
```

All sides are options which reside under the main accessory tag. The accessories available are shown in Table 7.

Number	Accessory	Num. of Sides	Robot
0	bigwheel	1	both
1	bridge	2	linkbot
2	caster	1	both
3	cube	5	linkbot
4	faceplate	1	linkbot
5	gripper	1	linkbot
6	l	3	mobot
7	omnidrive	4	linkbot
8	simple	2	both
9	smallwheel	1	both
10	square	4	mobot
11	tank	3	mobot
12	tinywheel	1	linkbot
13	wheel	1	both

Table 7: Robot Accessories

Multiple Sided Accessories
Accessories with multiple sides are configured with that number of `<side/>` tags as shown in this bridge example.

```
<bridge>
    <side id="1" robot="0" face="1"/>
    <side id="2" robot="1" face="3"/>
</bridge>
```

This will connect a bridge between robots with ids 0 and 1 on the robot's faces 1 and 3, respectively.

One-Sided Accessories
Accessories with only one side are configured with the robot and face options within the robot tag as shown in the wheel example below.

```
<smallwheel robot="1" face="3"/>
```

Daisy-Chaining
Since accessories of the Linkbot can be attached to each other and not just directly to the robot, there are options to daisy-chain the accessories together. To do this, the `face` option is replaced for a side with the `conn` option and the number corresponding to the accessory shown in the first column of Table 7. The example below shows the simple connector with a smallwheel attached to the third face of the robot.

```
<simple>
    <side id="1" robot="0" face="3"/>
    <side id="2" robot="0" conn="9"/>
</simple>
```

Custom Wheel Sizes
Custom wheel radii can be entered into the configuration file when using the `wheel` option. The radii of the preconfigured big, small, and tiny wheels are set internally to correspond to the produced wheels. Inputting a custom wheel radius is done through the `side` option when daisy-chaining a wheel. Below is a daisy-chained custom wheel with a radius of 0.001 meters.

```
<simple>
    <side id="1" robot="0" face="1"/>
    <side id="2" robot="0" conn="13" radius="0.001"/>
</simple>
```

A.6 Examples

Some example configuration files.

A.6.1 One Robot at (0, 0, 0)

```
<?xml version="1.0" encoding="UTF-8"?>
<config>
    <version val="1"/>
</config>
<graphics>
    <grid units="1" major="12" tics="1" minx="-48" maxx="48" miny="-48" maxy="48"/>
    <tracking val="1"/>
</graphics>
<sim type="0">
    <linkboti id="0">
        <position x="0" y="0" z="0"/>
        <rotation psi="0" theta="0" phi="0"/>
    </linkboti>
    <simple>
        <side id="1" robot="0" face="1"/>
        <side id="2" robot="0" conn="9"/>
    </simple>
    <simple>
        <side id="1" robot="0" face="3"/>
        <side id="2" robot="0" conn="9"/>
    </simple>
    <caster robot="0" face="2"/>
</sim>
```

A.6.2 Two Robots on X-Axis

```
<?xml version="1.0" encoding="UTF-8"?>
<config>
    <version val="1"/>
</config>
<graphics>
    <grid units="1" major="12" tics="1" minx="-48" maxx="48" miny="-48" maxy="48"/>
    <tracking val="1"/>
</graphics>
<sim>
    <linkboti id="0">
        <position x="0" y="0" z="0"/>
        <rotation psi="0" theta="0" phi="0"/>
    </linkboti>
    <simple>
        <side id="1" robot="0" face="1"/>
        <side id="2" robot="0" conn="9"/>
```

```
    </simple>
    <simple>
        <side id="1" robot="0" face="3"/>
        <side id="2" robot="0" conn="9"/>
    </simple>
    <caster robot="0" face="2"/>
    <linkboti id="1">
        <position x="0.1524" y="0" z="0"/>
        <rotation psi="0" theta="0" phi="0"/>
    </linkboti>
    <simple>
        <side id="1" robot="1" face="1"/>
        <side id="2" robot="1" conn="9"/>
    </simple>
    <simple>
        <side id="1" robot="1" face="3"/>
        <side id="2" robot="1" conn="9"/>
    </simple>
    <caster robot="1" face="2"/>
</sim>
```