

RoboSim User's Guide

Version 1.9.90

Copyright © July 26, 2015 by UC Davis C-STEM Center, All rights reserved.


Contents

1	Introduction	3
2	RoboSim GUI	3
2.1	Platform Selection	3
2.2	Add To Scene Selection	3
2.3	Scene View	4
2.4	Scene Configuration	4
2.4.1	General Configuration	4
2.4.2	Background	5
2.4.3	Robots	5
2.4.4	Obstacles and Drawings	6
3	Interacting with the GUI: Cheat Sheet	6
3.1	Adding a Robot	6
3.2	Deleting a Robot	6
3.3	Editing an Item	6
3.4	Colliding Objects	6
4	Running a Ch Program with RoboSim	6
5	Interacting with a RoboSim Scene	7
5.1	Keyboard Input	7
5.2	Mouse Input	9
6	Differences Between Virtual and Hardware Robots	9
A	Appendix: Manual Configuration File Generation	10
A.1	Header	10
A.2	Configuration Section	10
A.2.1	Version	10
A.2.2	Type	10
A.2.3	Grid	10
A.2.4	Tracking	10
A.2.5	Pause	10
A.2.6	Real Time	11
A.2.7	Mu	11
A.2.8	Coefficient of Restitution	11
A.3	Ground Section	11
A.3.1	Box	11
A.3.2	Cylinder	11
A.3.3	Sphere	12
A.4	Simulation Section	12
A.4.1	Robots	12
A.4.2	Accessories	15
A.5	Examples	16
A.5.1	One Robot at (0, 0, 0)	16
A.5.2	Two Robots on X-Axis	16
A.5.3	Explorer	17
B	Appendix: Background XML File Writing	19

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 the Barobo Linkbot and LEGO Mindstorms EV3 and NXT robots. 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  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. The RoboSim GUI is the place to configure the RoboSim Scene for interacting with the virtual robots. It is designed to be easy to use for configuring the Scene and the robots within it. There are four pieces of the GUI that hold four separate configurations.

- Platform Selection
- Add To Scene Selection
- Scene View
- Scene Configuration

Figure 1: The RoboSim GUI.

2.1 Platform Selection

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 **Virtual 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 Add To Scene Selection

The accordion along the left hand side holds all of the elements which can be added into the scene as shown in Figure 3.

Figure 3: Add To Scene: selection of items to add to Scene.

There are two groups: Robots and Obstacles and Drawings. The robots group has the list of robots available to be added into the scene which includes the Linkbot-I, Linkbot-L, Mindstorms EV3, and Mindstorms NXT. The Obstacles and Drawings group holds the extra elements for the scene. The difference between obstacles and drawings can be summarized as follows:

- Obstacles have mass and interact with the robots
- Drawings do not have mass and are only drawn on the screen

The possible obstacles are Box, Cylinder, and Sphere. The drawings are Lines, Points, and Text. To add a desired object to the scene, click and drag it onto the proper tab of the Scene Configuration. A drop indicator will show up when the object can be added as shown in Figure 4. Once the object has been added, it will show up in the Scene View.

Figure 4: Drop Indicator for adding a robot.

2.3 Scene View

The Scene View is a live version of the RoboSim Scene which shows the current state of the configuration. All keyboard and mouse interactions work as expected from the RoboSim Scene. Clicking a robot or obstacle selects it in yellow and makes it the current one in the Scene Configuration section to be able to edit its properties. As elements of the scene are changed, their state is automatically updated in the Scene View. Once ready to run code, the student starts the code and the RoboSim Scene will pop up showing the same state as shown in the GUI.

Since there are no limits placed on how items can be positioned in the Scene, it is a possibility that two or more can be configured to be overlapping. When this happens, all objects which overlap will be highlighted in red in the scene view and a message will appear for the user stating 'Objects are colliding'. To fix the problem, move one or more of the objects away from the others. As soon as RoboSim detects there is no longer an intersection, the message will disappear and the red highlighting will disappear.

2.4 Scene Configuration

The main section of the GUI is the Scene Configuration tab bar. Each tab holds one piece of the configuration parameters for the Scene.

2.4.1 General Configuration

The first one is the general configuration options. The parameters listed here affect the whole scene by changing how it looks and behaves.

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.

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.

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, either the US Customary defaults, shown in Figure 5, or the Metric defaults, as shown in Figure 6, will be set.

Figure 5: Default US Customary Grid Spacing.

Figure 6: Default Metric Grid Spacing.

2.4.2 Background

The background behind the robots can be changed to display different views for the Scene. There are four default options for the robots. Users can write their own and add them to the list if desired. Details on how to write a RoboSim Background are detailed in Appendix B. The four default backgrounds are None, RoboPlay 2014, RoboPlay 2015, and Outdoors. The 'None' background is a completely black scene with the robots moving through the empty space. The challenge boards from the 2015 and 2015 RoboPlay Challenge Competitions are included to easily play on the mats. The borders are solid obstacles so that the robots are not able to run off the sides. The 'Outdoors' background is the original RoboSim Scene consisting of the green grass and blue sky. Selecting each one automatically changes the background in the Scene View. The 'Add New Background' button allows users to import their own background folders.

2.4.3 Robots

All of the robots in the Scene are configured here. The Robot List (see Figure 7) holds all of the robots currently in the Scene. Robots are dragged from the 'Add To Scene' Selection List to this location to be included in the Scene.

Figure 7: Robot List: all robots in the Scene.

Clicking on one of the robots makes it the current one which means two things: it is highlighted yellow in the Scene View and the robot editor to the right is filled with the information about this robot. To the right is the Robot Editor dialog. All information about a robot that can be changed is done here. The possible options change depending upon the selected robot. Figure 8 shows the dialog for a Linkbot-I as an example.

Figure 8: Robot Editor: editor for a Linkbot-I.

The dropdown allows the user to change the type of robot which in turn changes the options. A name for the robot can be entered if desired. The X and Y positions of the robot can be changed either through the arrows or typing in a desired number. The robot angle is measured from the X-axis rotating in a counter-clockwise direction. Left and right wheels can be added separately to the robot. The options are all of the sizes of wheels made for the Linkbots. Wheel size options for the Mindstorms robots are the two which are distributed with the robot (1.1 and 1.6 inch radii). The LED color button brings up a color picker dialog

to select any desired color. The buttons at the bottom remain the same for each robot. The Previous and Next buttons allow the previous or next robot in the list to become the one selected for editing. The Delete button deletes the robot from the Scene.

2.4.4 Obstacles and Drawings

The obstacles and drawings tab is organized in an identical manner to the Robots tab. A list of the objects in the scene are on the left. New ones are dragged from the left of the GUI into this list. An editor for each object is on the right. The information in the editor has a few general sections which change slightly depending upon the specific item being edited. The physical properties of the object, such as the length, width, and height of a box or the radius of the sphere, the object's position in space in X, Y, and Z directions, mass of the object, and the color.

3 Interacting with the GUI: Cheat Sheet

3.1 Adding a Robot

The list of robots which can be added to the Scene are under the robots group of the 'Add To Scene' accordion as shown in Figure 3. The list of robots in the scene is in the Robots tab of the Scene Configuration. Click and drag a robot from the left to the right and drop it onto the list. This add it into the Scene. The robot should now be in the Robot List as well as drawn in the Scene View.

3.2 Deleting a Robot

To delete a robot already in the Scene, it first has to be selected to become the current robot. This is shown in two ways: it is highlighted yellow in the Scene View and is selected blue in the Robot List. To select a robot, it can be clicked in either of these locations. Deleting can be accomplished by either pressing the delete key on the keyboard or the delete button in the robot editor.

3.3 Editing an Item

If a robot is selected, its information can be edited. Selecting can be done by clicking either the robot in the Scene View to highlight it or in the robot list. Once the robot is selected, its information is loaded into the editor to the right of the Robot List. Changing any of this information will be reflected immediately in the Scene View.

3.4 Colliding Objects

Since there are no limits placed on how items can be positioned in the Scene, it is a possibility that two or more can be configured to be overlapping. When this happens, all objects which overlap will be highlighted in red in the scene view and a message will appear for the user stating 'Objects are colliding'. To fix the problem, move one or more of the objects away from the others. As soon as RoboSim detects there is no longer an intersection, the message will disappear and the red highlighting will disappear.

4 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 9 will be displayed.

Paused: Press any key to start

is displayed in the RoboSim scene to remind 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 9: 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 10.

Figure 10: 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.

5 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.

5.1 Keyboard Input

The RoboSim scene responds to keyboard input as outlined in Table 1. 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
C	center the current robot in the View (GUI only)
N	toggle grid line numbering
R	toggle robot visibility and enable tracking
T	toggle robot tracking
any other key	Pause and resume the simulation

Table 1: 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 second quadrant. This view can be seen in any of the RoboSim scene screenshots within this document, except for Figure 11 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.

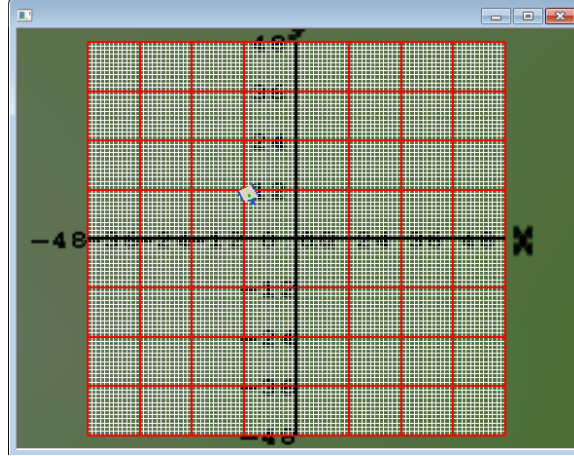


Figure 11: A RoboSim scene with the overhead viewing angle.

In the RoboSim GUI the highlighted robot is the one which is being edited currently. To center this robot in the Scene View, press the 'c' key. This will move the camera to put this robot at the center of the view.

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 12 shows a RoboSim scene with a tracked robot trajectory only.

Figure 12: 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.

5.2 Mouse Input

Clicking on a robot in a RoboSim scene will enable a pop up which displays the robot's name, if set in the RoboSim GUI, the robot number, and the current position of the robot, as shown in Figure 13. Clicking again the displayed position for the robot will disappear.

Figure 13: 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 2. 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 2: Mouse input for the RoboSim scene.

6 Differences Between Virtual and Hardware Robots

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 Appendix: 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 Type

```
<type val="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 `<sim></sim>` are individual. Any number above that represents the preconfigured robots within the GUI.

A.2.3 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.2.4 Tracking

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

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

A.2.5 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.2.6 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.2.7 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.2.8 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 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. Everything to be added is put between the `<ground>` tags.

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

A.3.1 Box

```
<box mass="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, size, position, and rotation parameters. Mass is given as the total mass for the object. 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">
  <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, 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 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">
  <size radius="1"/>
  <position x="1" y="1" z="1"/>
</sphere>
```

The ground sphere is configured with the mass, size and position parameters. Mass is the total mass of the object. 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 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.4.1 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 3. 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.

<code><linkboti/></code>
<code><linkbotl/></code>
<code><linkbott/></code>
<code><mobot/></code>

Table 3: 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.

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

Table 4: 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 5: 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. `<name>` specifies the name of the robot. `<joint>` allows the joints to be rotated initially as opposed to being built at zero rotation. `<wheels>` creates a car robot by attaching wheels and a caster to each robot. The numbers of each wheel correspond to the connectors listed in Table 6. Examples are shown below.

```

<linkboti id="0">
  <position x="0" y="0" z="0"/>
  <rotation psi="0" theta="0" phi="0"/>
<name>Kevin</name>
  <joint f1="-20" f2="0" f3="20"/>
<wheels left="1" right="2"/>
</linkboti>

<mobot id="0">
<name>Kevin</name>
  <joint f1="-20" f2="0" f3="20"/>
  <position x="0" y="0" z="0"/>
  <rotation psi="0" theta="0" phi="0"/>
  <joint a1="-20" a2="0" a3="20" a4="45"/>
<wheels left="1" right="2"/>
</mobot>

<ev3 id="0">
<name>Kevin</name>
  <position x="0" y="0" z="0"/>
  <rotation psi="0" theta="0" phi="0"/>
  <joint w1="-20" w2="20"/>
<wheels left="1" right="2"/>
</ev3>

<nxt id="0">
<name>Kevin</name>
  <position x="0" y="0" z="0"/>
  <rotation psi="0" theta="0" phi="0"/>

```

```

    <joint w1="-20" w2="20"/>
<wheels left="1" right="2"/>
</nxt>

```

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 14. 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 15, 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.

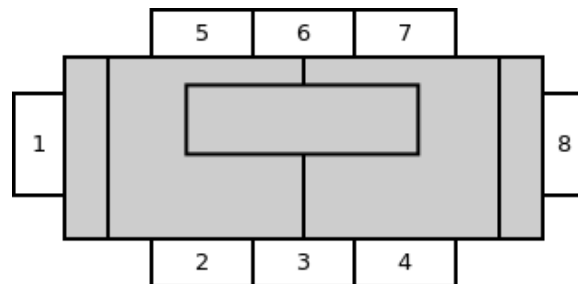


Figure 14: Mobot Connection Locations.

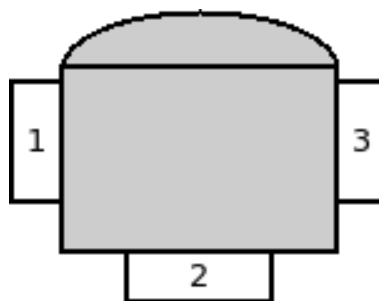


Figure 15: Linkbot Connection Locations.

A.4.2 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 6.

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 6: 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 6. 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.5 Examples

Some example configuration files.

A.5.1 One Robot at (0, 0, 0)

```

<?xml version="1.0" encoding="UTF-8"?>
<config>
  <version val="1"/>
  <type val="0"/>
<grid units="1" major="12" tics="1" minx="-48" maxx="48" miny="-48" maxy="48"/>
  <tracking val="1"/>
</config>

<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"/>
</sim>

```

A.5.2 Two Robots on X-Axis

```

<?xml version="1.0" encoding="UTF-8"?>
<config>
  <version val="1"/>
  <type val="0"/>
<grid units="1" major="12" tics="1" minx="-48" maxx="48" miny="-48" maxy="48"/>

```



```

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

<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>

```

A.5.3 Explorer

```

<?xml version="1.0" encoding="UTF-8"?>
<config>
  <version val="1"/>
  <type val="1"/>
  <grid units="1" major="12" tics="1" minx="-48" maxx="48" miny="-48" maxy="48"/>
  <tracking val="1"/>
</config>

<sim>
  <linkboti id="0">
    <position x="0" y="0" z="0"/>
    <rotation psi="0" theta="0" phi="-90"/>
  </linkboti>
  <linkboti id="1"/>
  <linkboti id="2" orientation="3">
    <joint f1="-20" f2="0" f3="20"/>
  </linkboti>
  <linkboti id="3">

```

```

        <joint f1="-90" f2="0" f3="90"/>
</linkboti>
<linkbotl id="4" orientation="3"/>
<simple>
    <side id="1" robot="0" face="3"/>
    <side id="2" robot="0" conn="9"/>
</simple>
<simple>
    <side id="1" robot="1" face="1"/>
    <side id="2" robot="1" conn="9"/>
</simple>
<cube>
    <side id="1" robot="0" face="1"/>
    <side id="2" robot="0" conn="2"/>
    <side id="3" robot="1" face="3"/>
    <side id="5" robot="2" face="2"/>
</cube>
<bridge>
    <side id="1" robot="2" face="1"/>
    <side id="2" robot="3" face="1"/>
</bridge>
<bridge>
    <side id="1" robot="2" face="3"/>
    <side id="2" robot="3" face="3"/>
</bridge>
<simple>
    <side id="1" robot="3" face="2"/>
    <side id="2" robot="4" face="2"/>
</simple>
    <gripper robot="4"/>
</sim>

```

B Appendix: Background XML File Writing

hi