

[← Back](#)[\(/tutorials?cat=guided_b\)](/tutorials?cat=guided_b)

Beginner: Model Editor

Model Editor

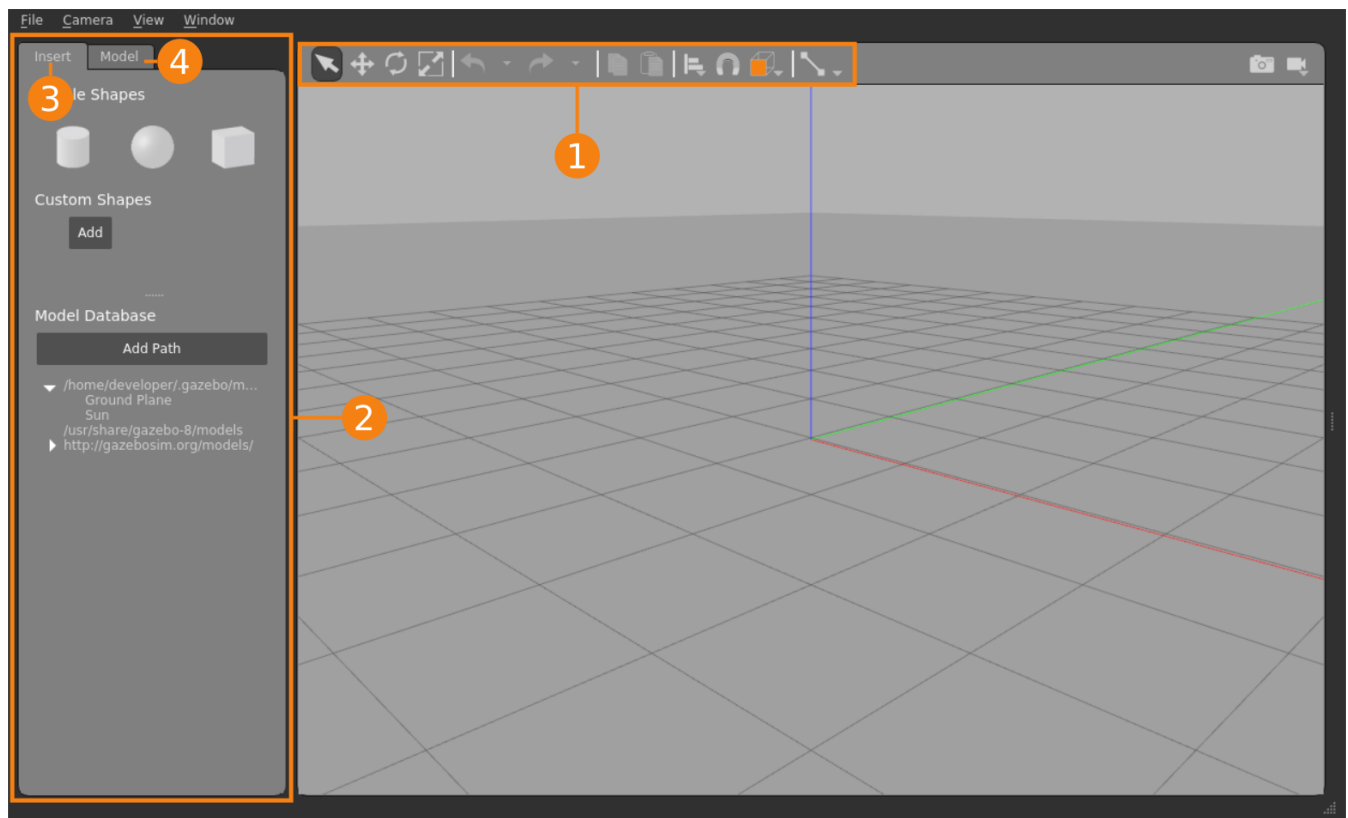
Now we'll construct our simple robot. We'll make a wheeled vehicle and add a sensor that allows us to make the robot follow a blob (person).

The Model Editor lets us construct simple models right in the Graphical User Interface (GUI). For more complex models, you'll need to learn how to write SDF (<http://sdformat.org/>) files, and check out the tutorials on building a robot (http://gazebo-sim.org/tutorials?cat=build_robot). But for now, we can do everything right in the Gazebo GUI!

Model Editor User Interface

To enter the Model Editor, click on **Edit** in the menu bar and select **Model Editor**. Or, use the hotkeys, Ctrl+M. **Physics and the simulation will be paused** as soon as you are in the Model Editor.

The Model Editor interface looks similar to the main Gazebo UI but with some subtle differences. The left panel and top Toolbar now contain only widgets for editing and creating parts of the model. The bottom Toolbar that displays simulation data is hidden since the simulation is now paused.



1. **Toolbar** - Contains tools for editing the model
2. **Palette** - Also known as *Left Panel*. Has two tabs for editing the model.

3. **Insert tab** - Tools for adding links and nested models
4. **Model tab** - Allows editing model properties and contents

Palette (Left Panel)

The **Palette** has two tabs.

- **INSERT**: The Insert tab is where you add new parts (links and models) to the Model Editor. There are three sections.
 - Simple Shapes: These are primitive geometries that can be inserted to form a *link* in the model.
 - Custom Shapes: The **Add** button allows you to import custom meshes to from a *link* in the model. It currently supports COLLADA (.dae), 3D Systems (.stl), Wavefront (.obj) and W3C SVG (.svg) files.
 - Model Database: Has a list of *models*. These can be inserted into the Model Editor in the same way as simple shapes. Once inserted, they are called *Nested Models*.
- **MODEL**: The Model tab allows you to set the name and basic parameters of the model you are building. It displays a list of the links, joints, nested models, and plugins that are in the model. The parameters can be modified using the *Link Inspector*. It can be opened using any of these methods.
 1. double-clicking on the item in the list
 2. double-clicking on the item in the Scene
 3. right-clicking on the item in the list and selecting **Open Link Inspector**
 4. right-clicking on the item in the Scene and selecting **Open Link Inspector**

Toolbar

Like in Simulation mode, the main Toolbar in the Model Editor includes tools for interacting with the objects in the Scene (see the User Interface (http://gazebo.org/tutorials?cat=guided_b&tut=guided_b2) tutorial).

The available tools are selection, translation, scaling, rotation, undo & redo, copy & paste, alignment, snapping, view adjustment, and joint creation.

Limitations

The Model Editor supports most of the basic model building tasks that can be done by writing SDF. However, there are a few features that are not yet available:

- editing nested models and links within nested models.
- adding and editing certain geometry types including Plane and Polyline.
- support for heightmaps.
- CAD functionalities.

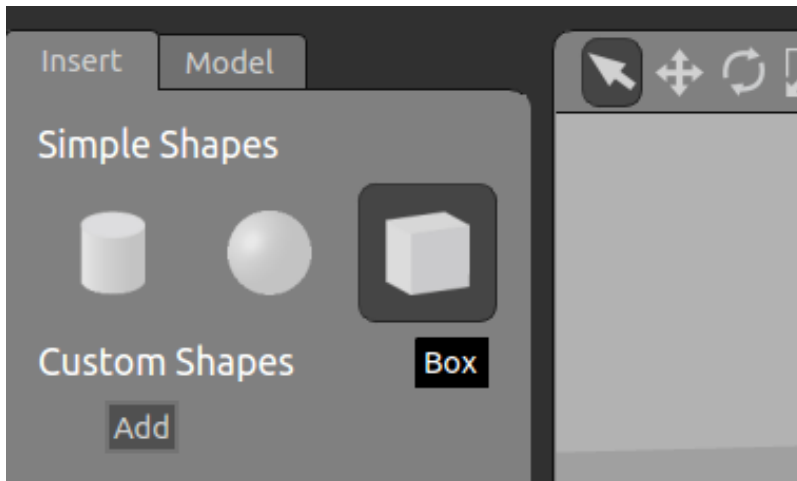
Vehicle construction

Creating a vehicle

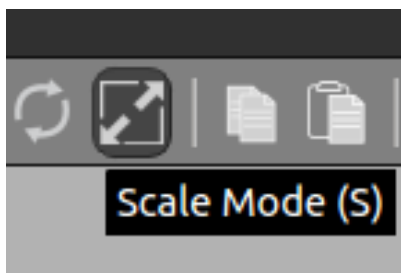
This section provides step-by-step instructions on creating a simple vehicle model in the Model Editor.

Chassis

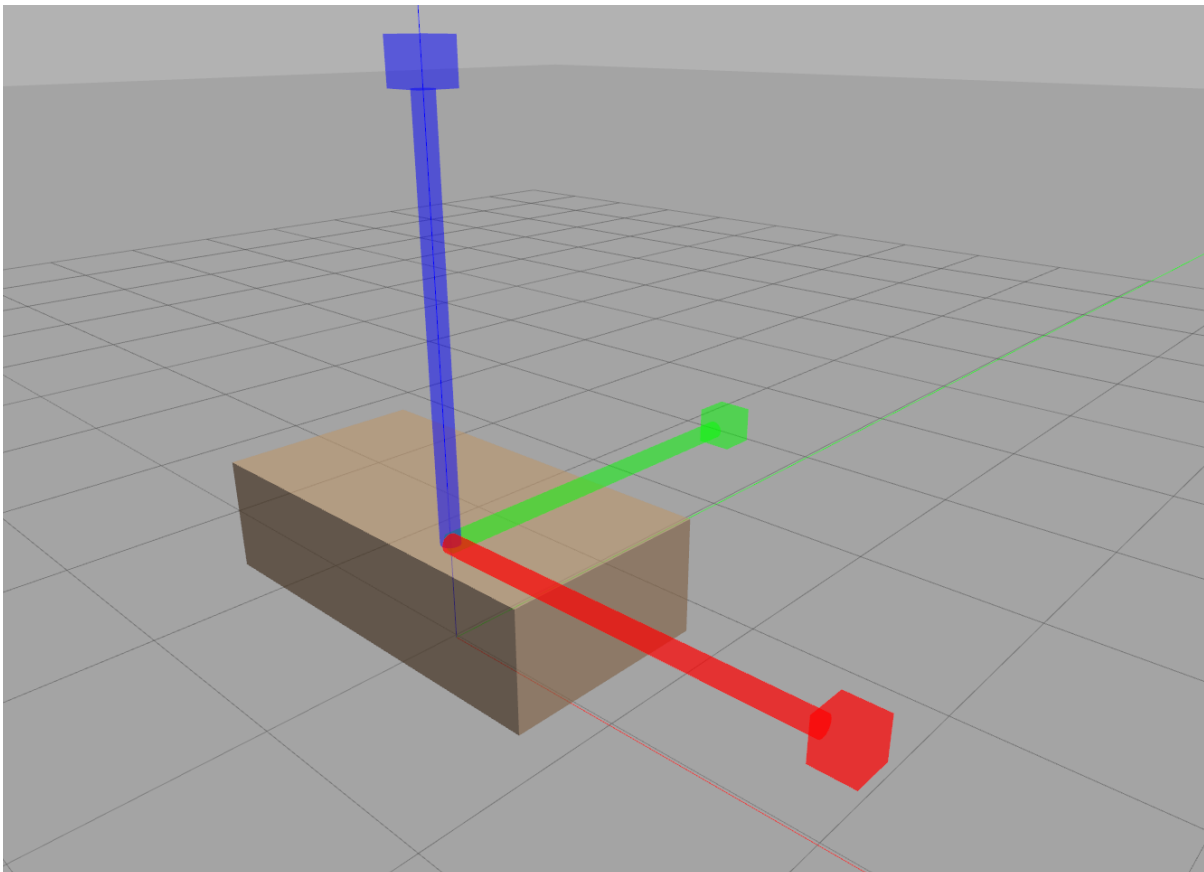
1. First, we'll create the vehicle chassis. In the Insert tab in the left panel, click once on the Box icon, move the cursor to anywhere in the Scene, and click again to release the box.



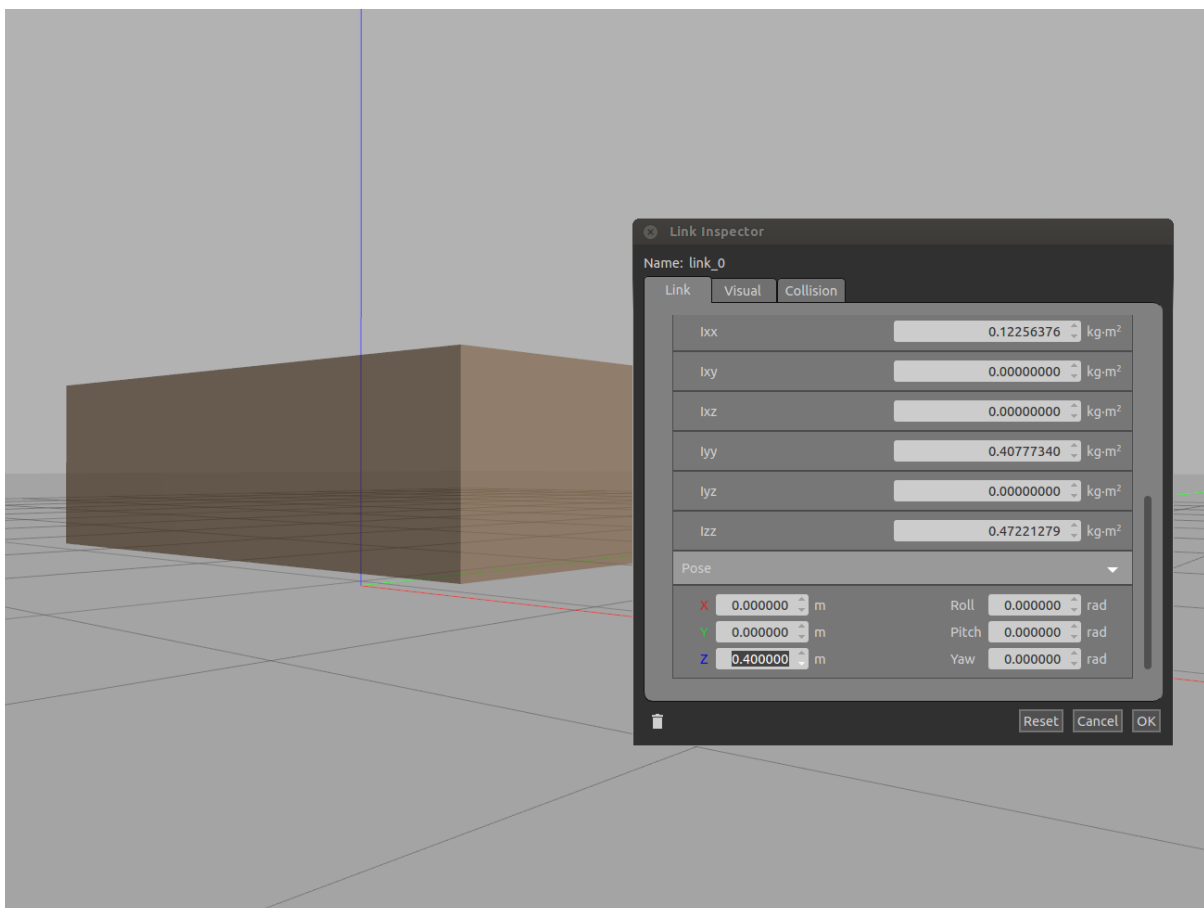
2. Next, resize the box so that it looks more like the shape of a car chassis. We can do this by selecting the Scale tool located on the top Toolbar. Select the box in the Scene, and a RGB-colored marker should appear over the box. The red marker represents the X axis, green is Y, and blue is Z. Move the mouse over the red marker to highlight it, then click and drag to make the chassis longer along the X axis. Scale the chassis so it is roughly 2 meters long. You can estimate this by looking at the 1x1 meter grids on the ground.



3. Now flatten the chassis with the Scale tool. Click and drag the blue marker down so that the chassis is approximately half of its original size.

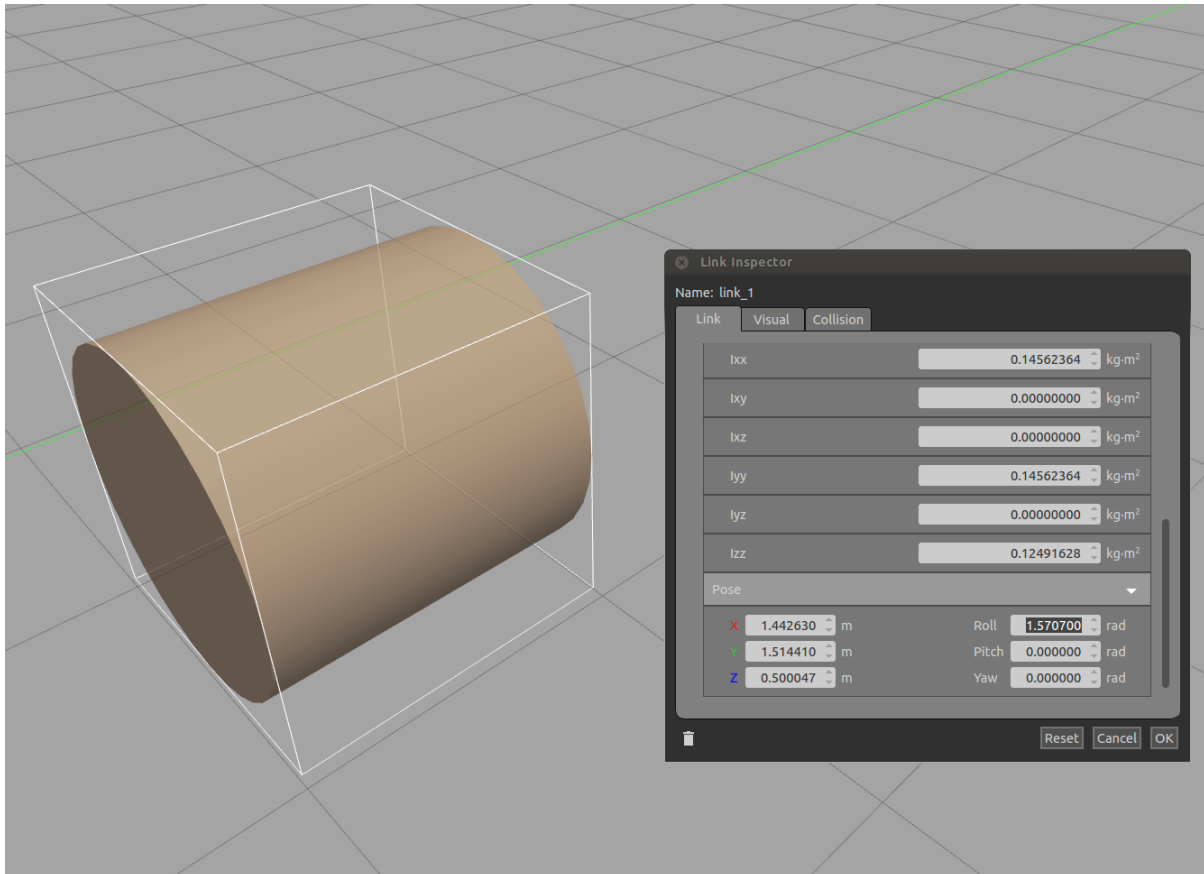


4. We want to lower the chassis closer to the ground. To give exact measurements, we will use the Link Inspector. Double-click on the box to bring up the Inspector. Scroll down to the bottom of the Link tab to find the **Pose** parameters and change **z** to be 0.4m then click outside the box (don't hit enter). Click **OK** to save the changes and close the Inspector.

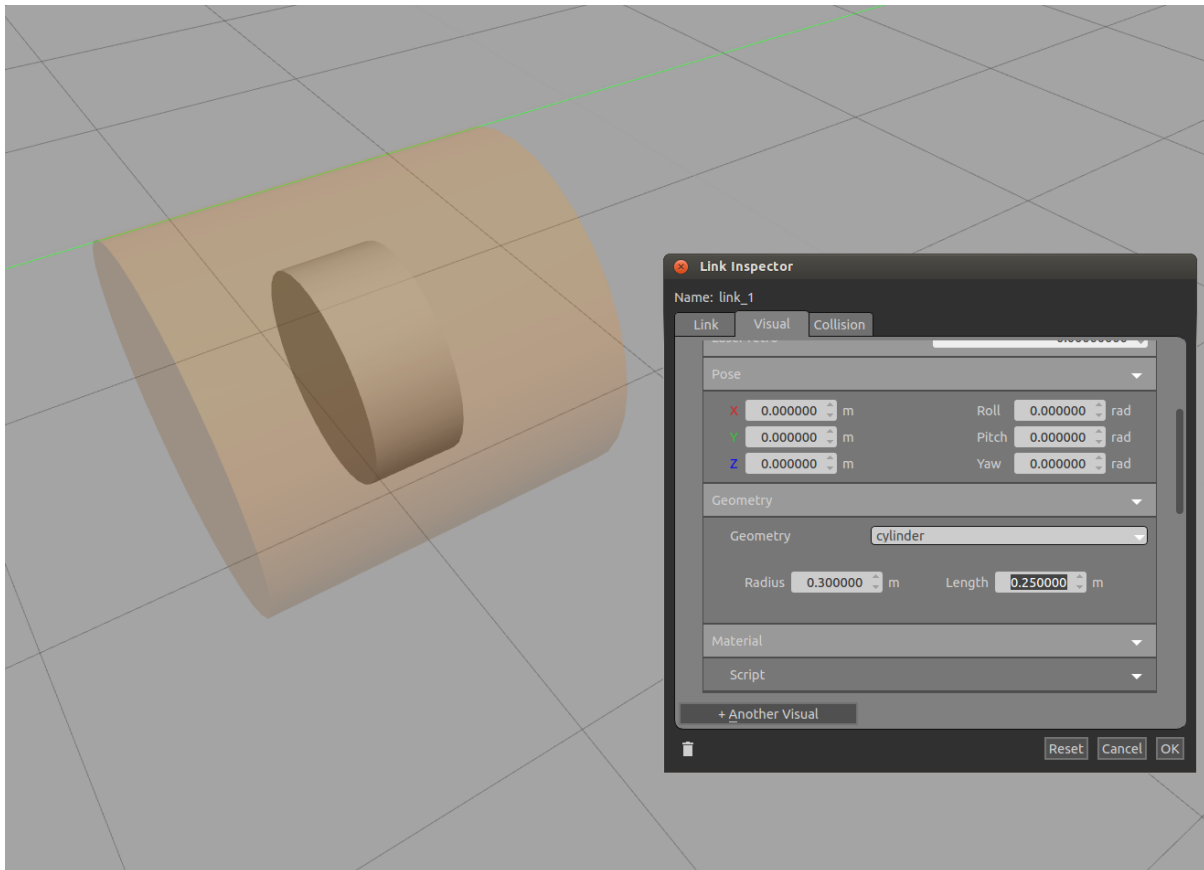


Front Wheels

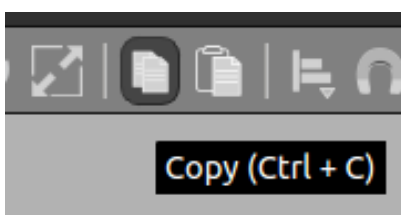
1. Let's move on to the front wheels. Start by inserting a cylinder from the Insert tab on the left panel.
2. The cylinder in its default orientation will not roll very well. Let's rotate it along the X axis using the Link Inspector. Double-click on the cylinder, scroll to the Pose section at the bottom, and change **Roll** to 1.5707 radians (90 degrees) and click outside the box. Do not close the Inspector just yet.



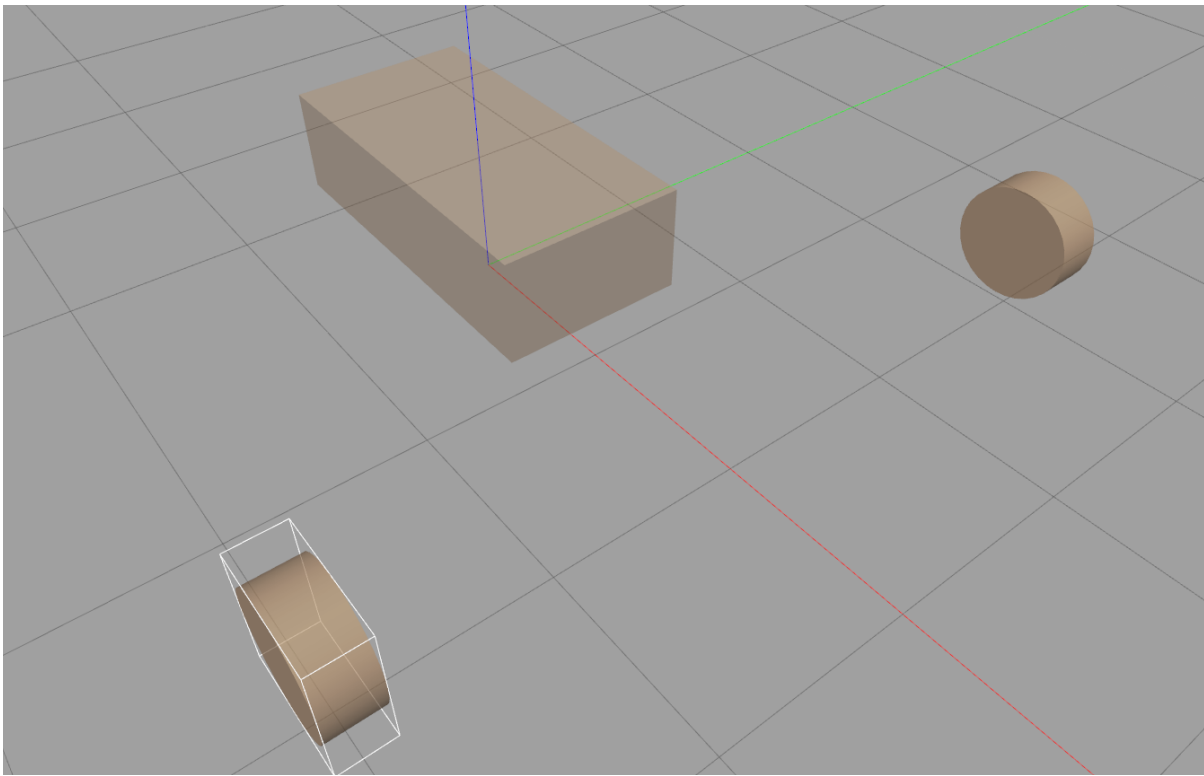
3. Next, resize the wheel by giving it exact dimensions. Go to the Visual tab to see the list of visuals in this link. There should only be one. Expand the visual item by clicking on the small arrow next to the **visual** text label. Scroll down to the **Geometry** section and change the **Radius** to 0.3m and **Length** to 0.25m.



4. You should now see a smaller cylinder inside a bigger cylinder. This is expected as we have only changed the visual geometry but not the collision. A 'visual' is the graphical representation of the link and does not affect the physics simulation. On the other hand, a 'collision' is used by the physics engine for collision checking. To also update the wheel's collision, go to the Collision tab, expand the only collision item, and enter the same Geometry dimensions. **Radius** : 0.3m and **Length** : 0.25m. Click on **OK** to save the changes and close the Inspector.
5. Now that we have created our first wheel, we'll use it as a template and make another one. Select the wheel and click on the Copy icon in the top Toolbar.

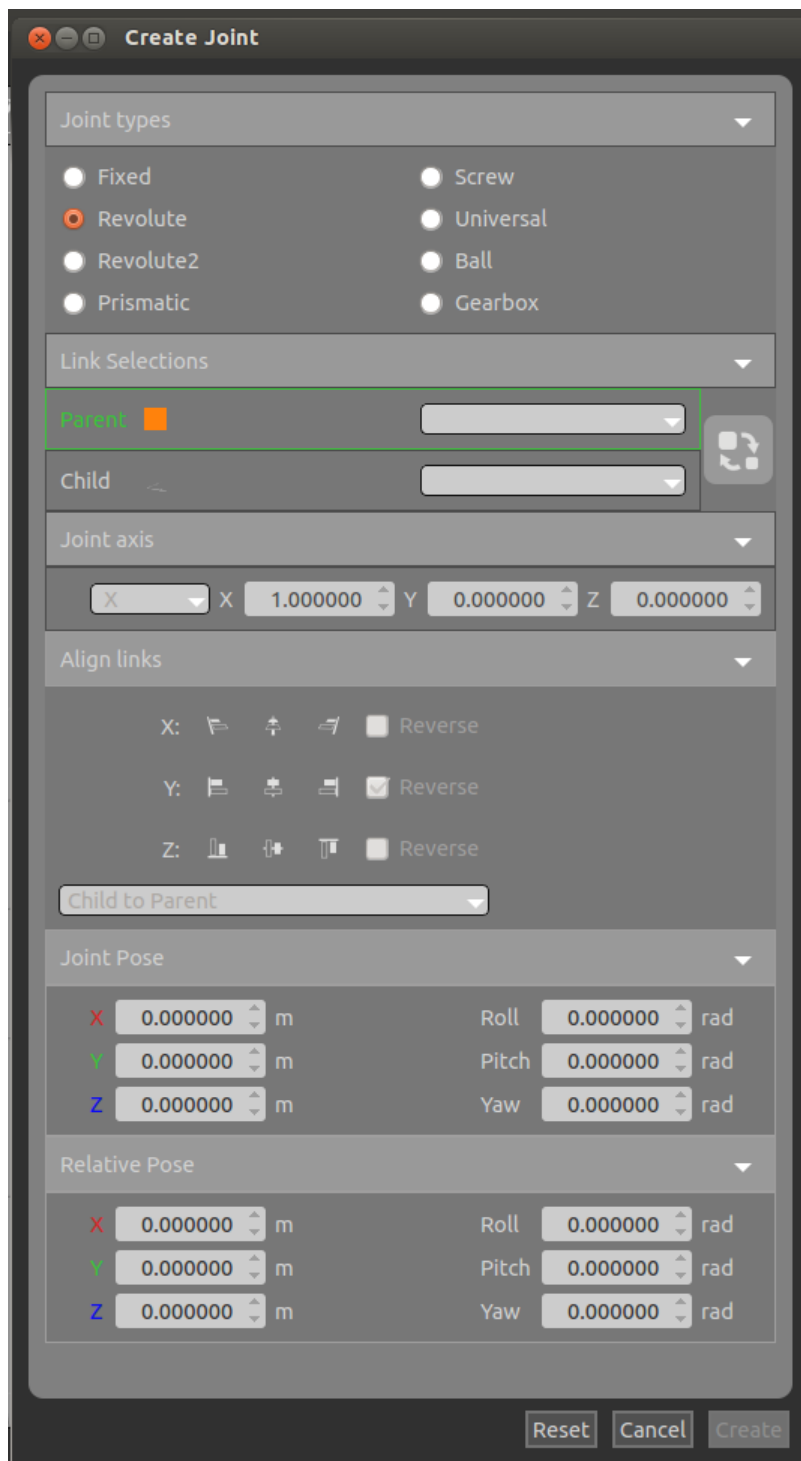


6. Click on the Paste icon and move the mouse back to the Scene to insert the copy.



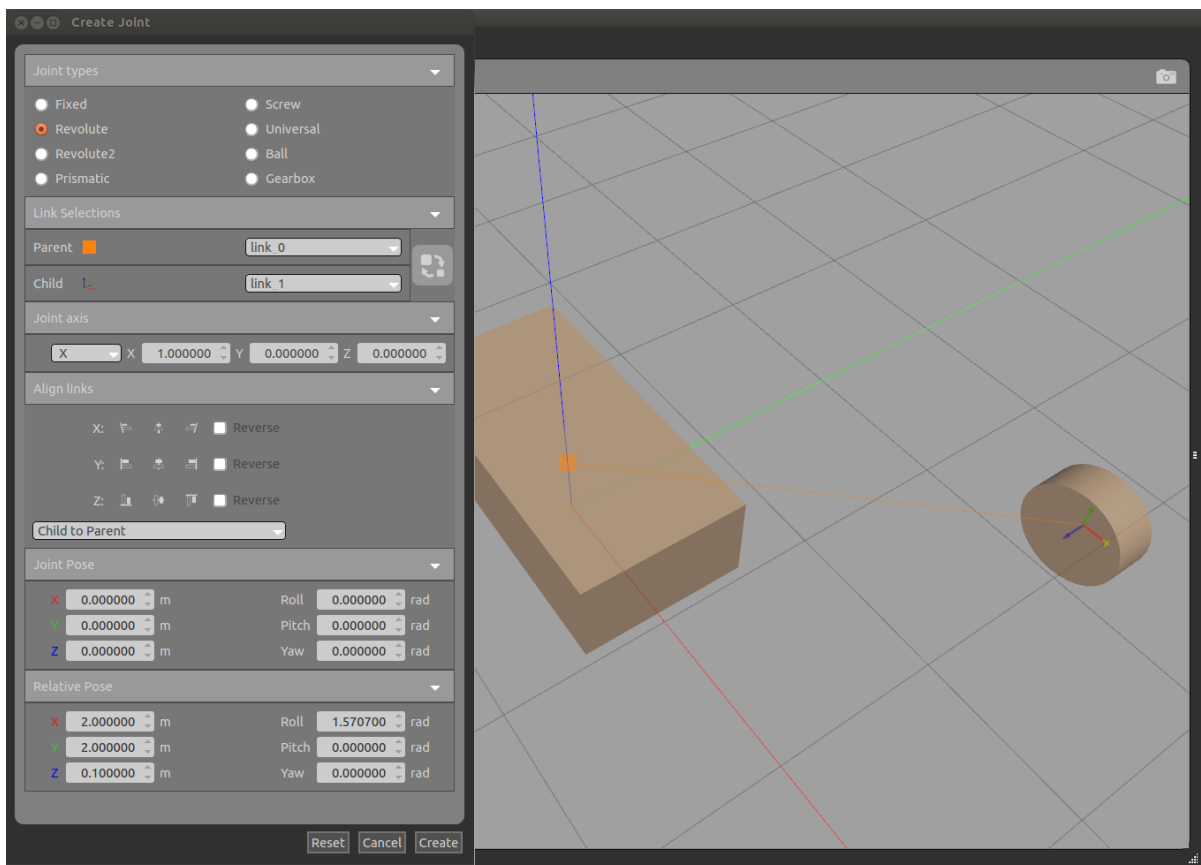
Let's now make sure that the vehicle will drive correctly by aligning the chassis along the positive X axis (the red marker in the scene). When you add the wheels in the next step, make sure they are at the end of the vehicle that is extending along the positive X axis.

7. The chassis and the wheels are currently free-moving bodies. To constrain their motion, we'll add joints between each wheel and the chassis. Click on the Joint icon in the top Toolbar to bring up the Joint Creation dialog.

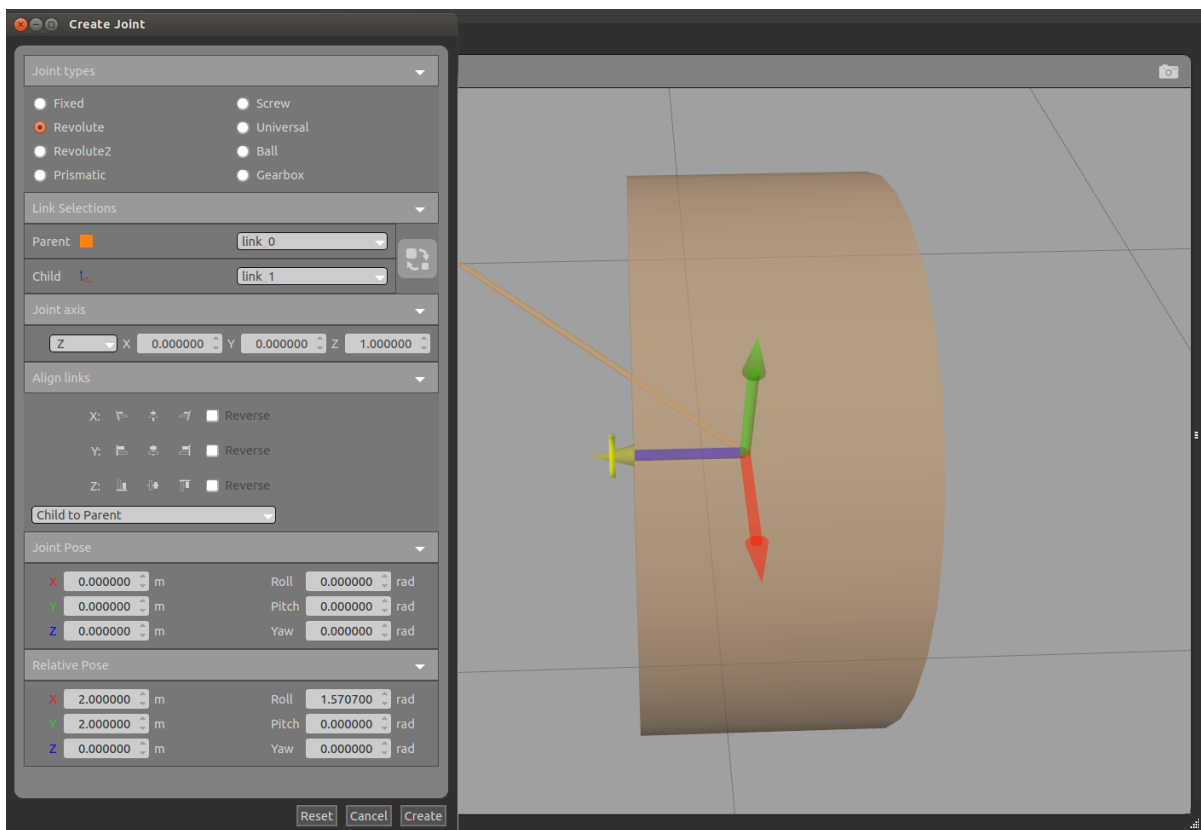


8. The Joint Creation dialog contains joint properties that are commonly specified for a joint. Before configuring any of the properties, you are prompted to select the parent and child links of the joint. Move the mouse over the chassis in the Scene to see it highlighted, and click on it to set it as the parent of the joint.
9. Move the mouse to the left front wheel; a line should now extend from the origin of the chassis to the end of the mouse. Click on the wheel to set it as the child of the joint. A new joint is created. By default it is a revolute joint (as indicated in the **Joint Types** section in the dialog) which just happens to be the joint type we want.

Note: You may find it useful to change the view angle at this point. This can be done in the Upper Toolbar; click the cube icon with an orange side.

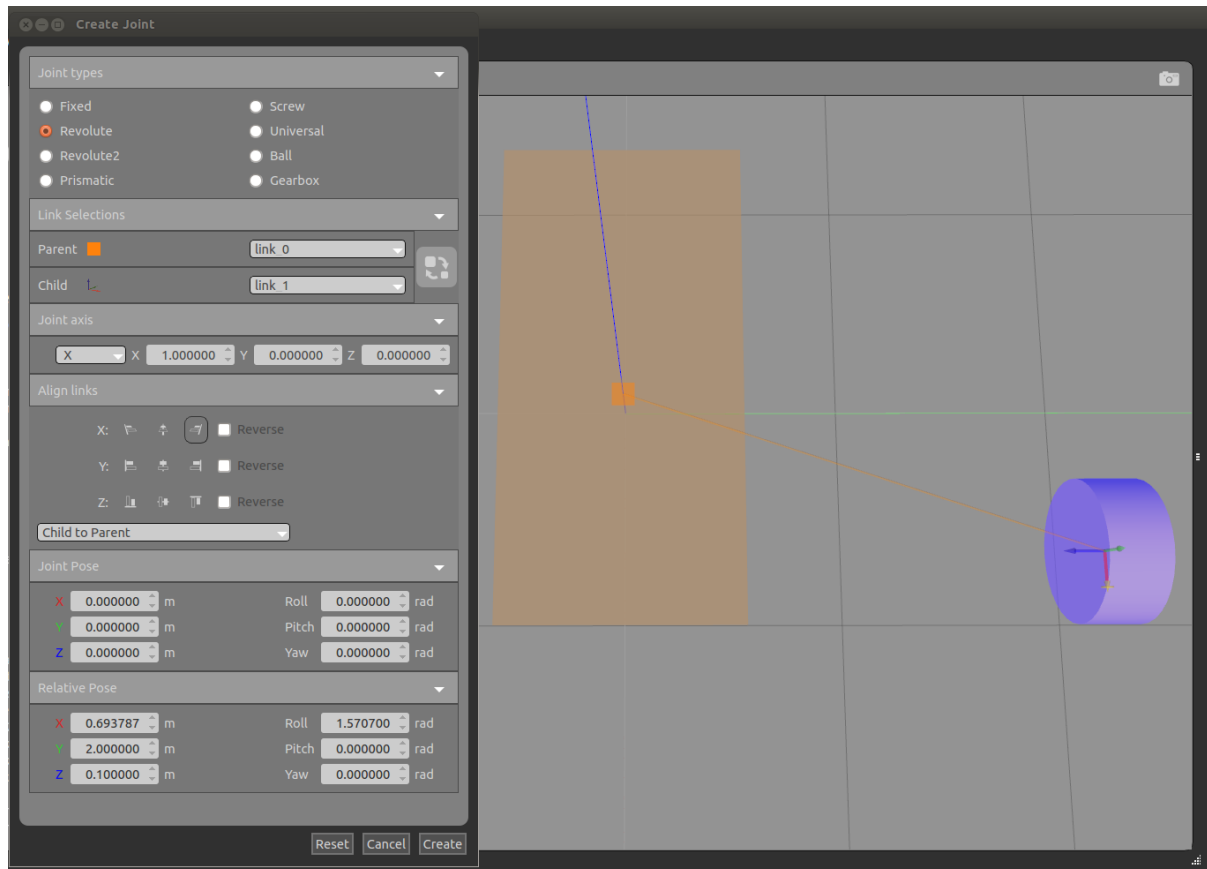


10. Next, we need to configure the axis of rotation of the wheel. In the Joint Creation dialog, find the **Joint axis** section and change the axis to be **z** (0, 0, 1). Pay attention to the RGB joint visual on the wheel. You should see that a yellow ring now appears over the blue arrow of the joint visual to indicate that it is the axis of rotation.

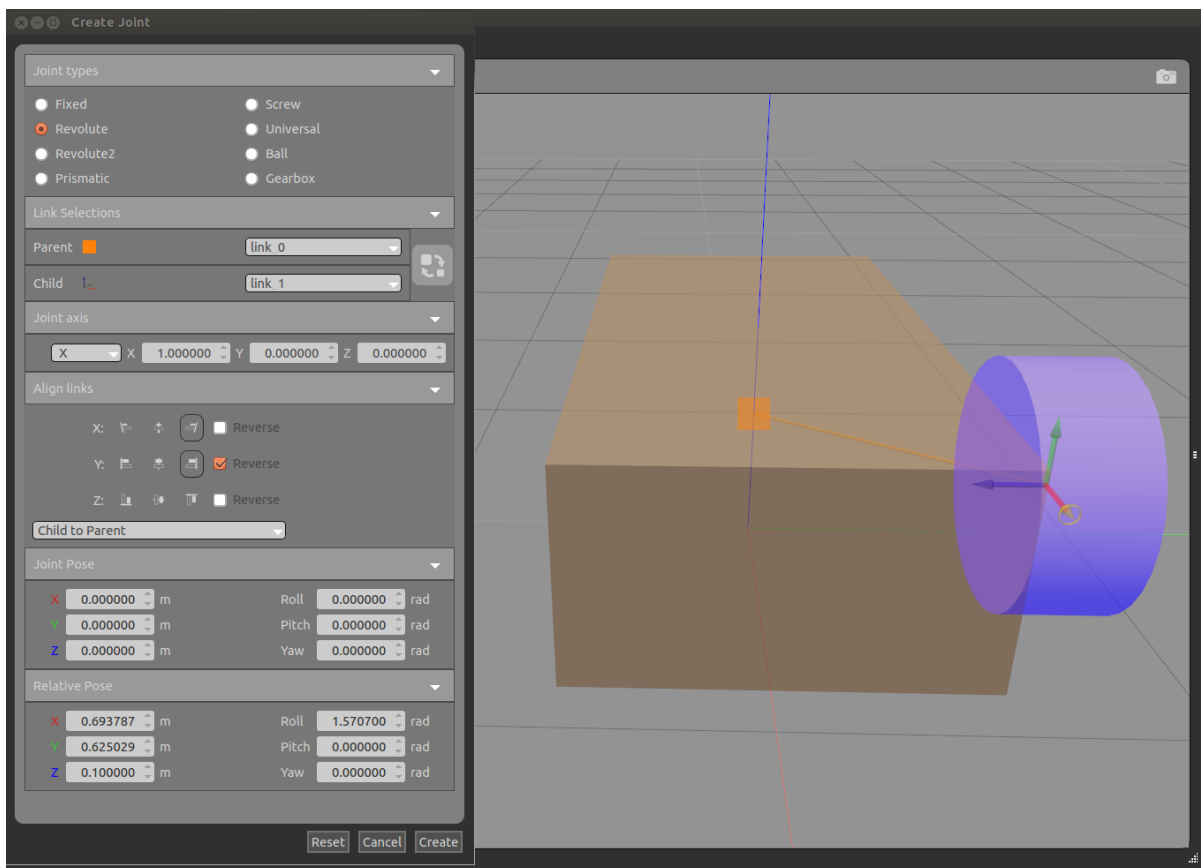


11. To align the wheel next to the chassis, we will use the different alignment options in the **Align links** section in the Joint Creation dialog. First we will align in the X axis, so click on the **x Align Max** option to see the result of the alignment. The cylinder should be

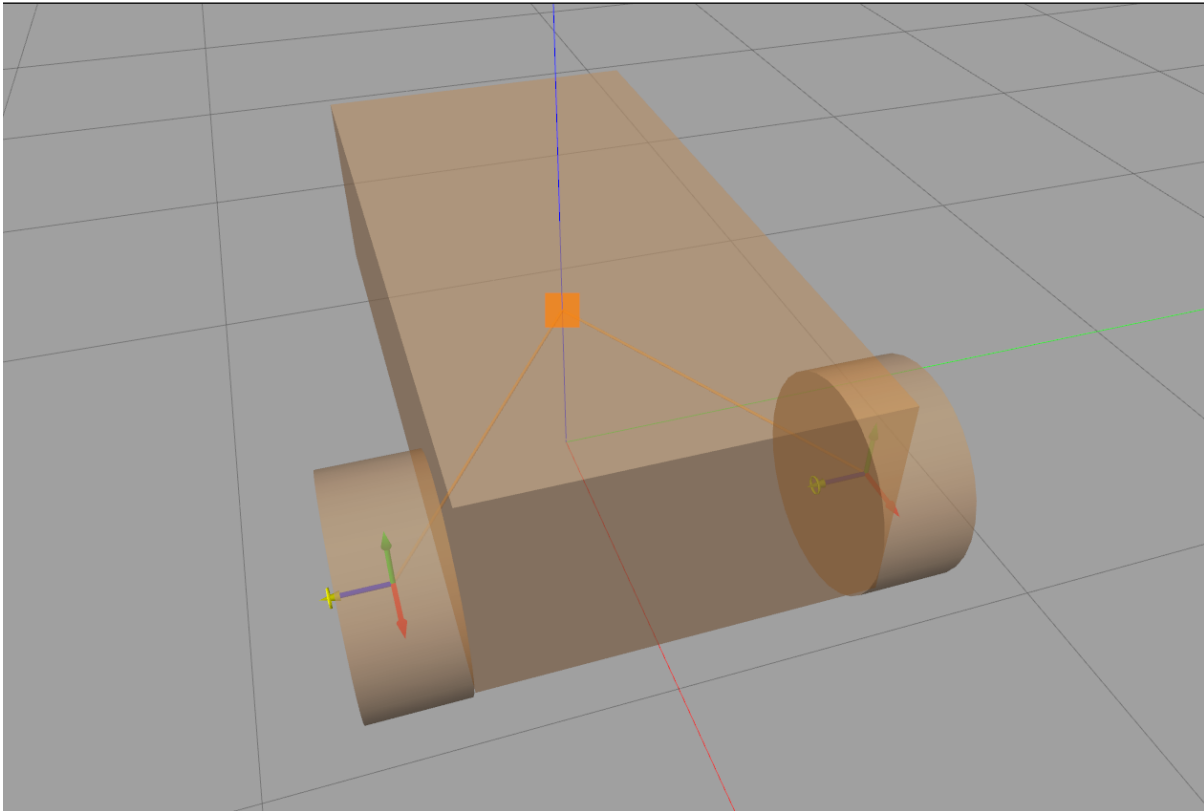
highlighted to indicate that its pose has changed.



12. In our example, we want to position the wheel flush against the chassis. To bring the wheel closer, click the **Y Align Max** option. However, it is not quite what we want yet. Click the **Reverse** option next the Y alignment options to align the wheel's minimum (reverse of maximum) to the chassis's maximum. Note that the **Reverse** option is applied to the child link since the default alignment configuration shown in the drop down list below is **Child to Parent**. If **Parent to Child** configuration is set, the **Reverse** option will be applied to the parent link. Press **Create**.

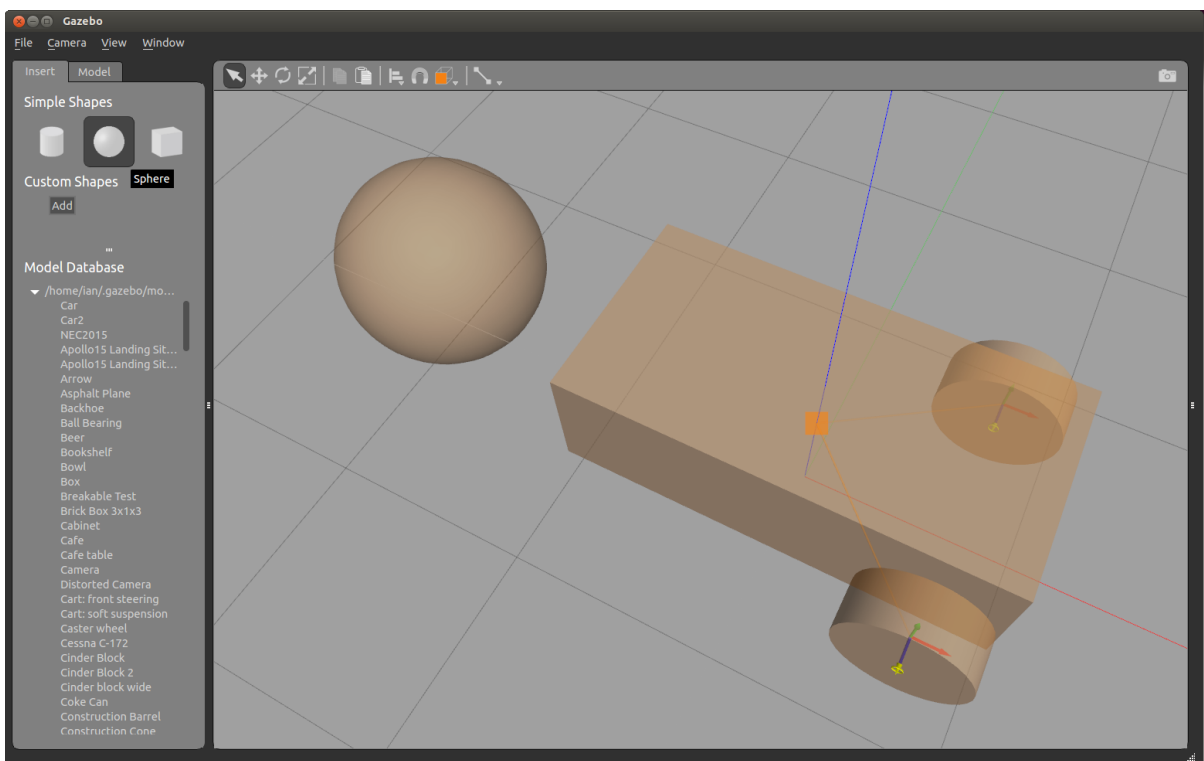


13. To position the wheel above the ground, open the Link Inspector by double-clicking on the wheel. We can use the **Pose** section at the bottom of the dialog to move the wheel. Given that the wheel has a radius of 0.3m, go ahead and change the **z** position to 0.3m to place on the ground, and press **Ok**.
14. Repeat the joint creation process and axis configuration for the other front wheel, make sure that a) the chassis is the parent of the joint and the wheel is the child, b) the axis of rotation is set to **z**, and c) use the **Y Align Min** option to align the right wheel as it is on the other side of the chassis.

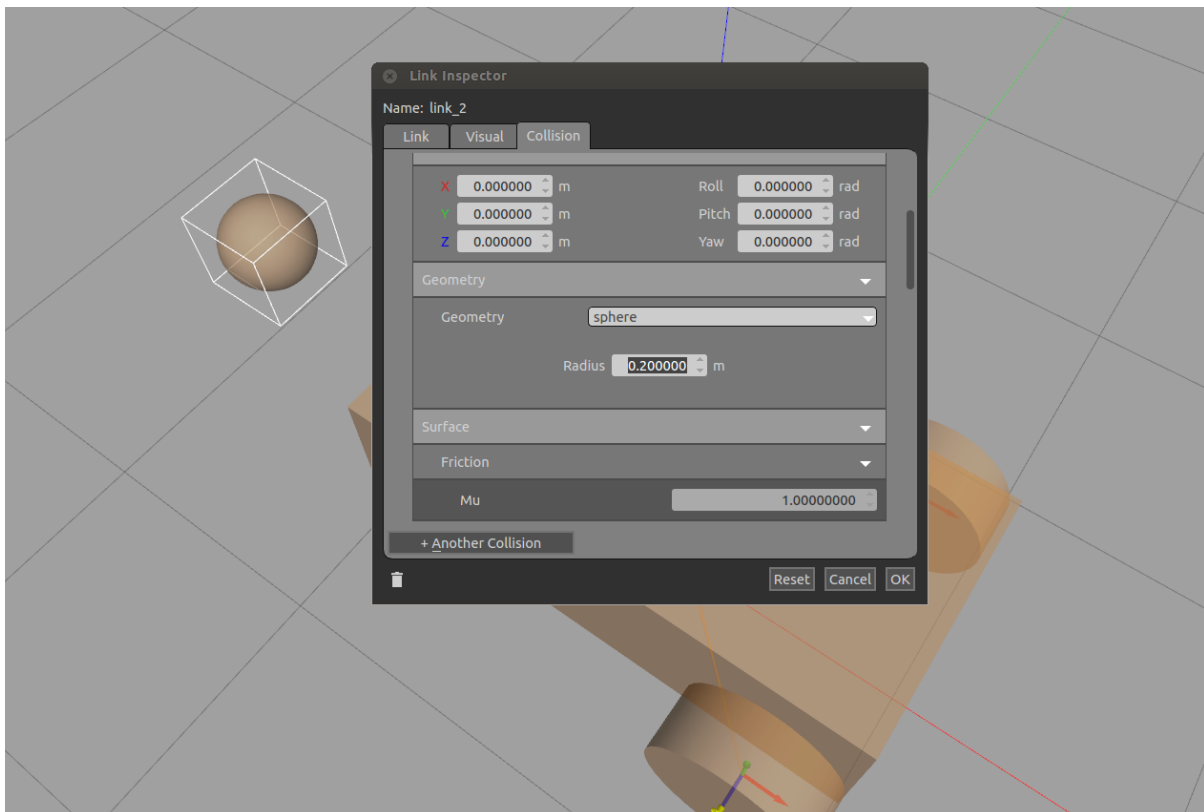


Caster Wheel

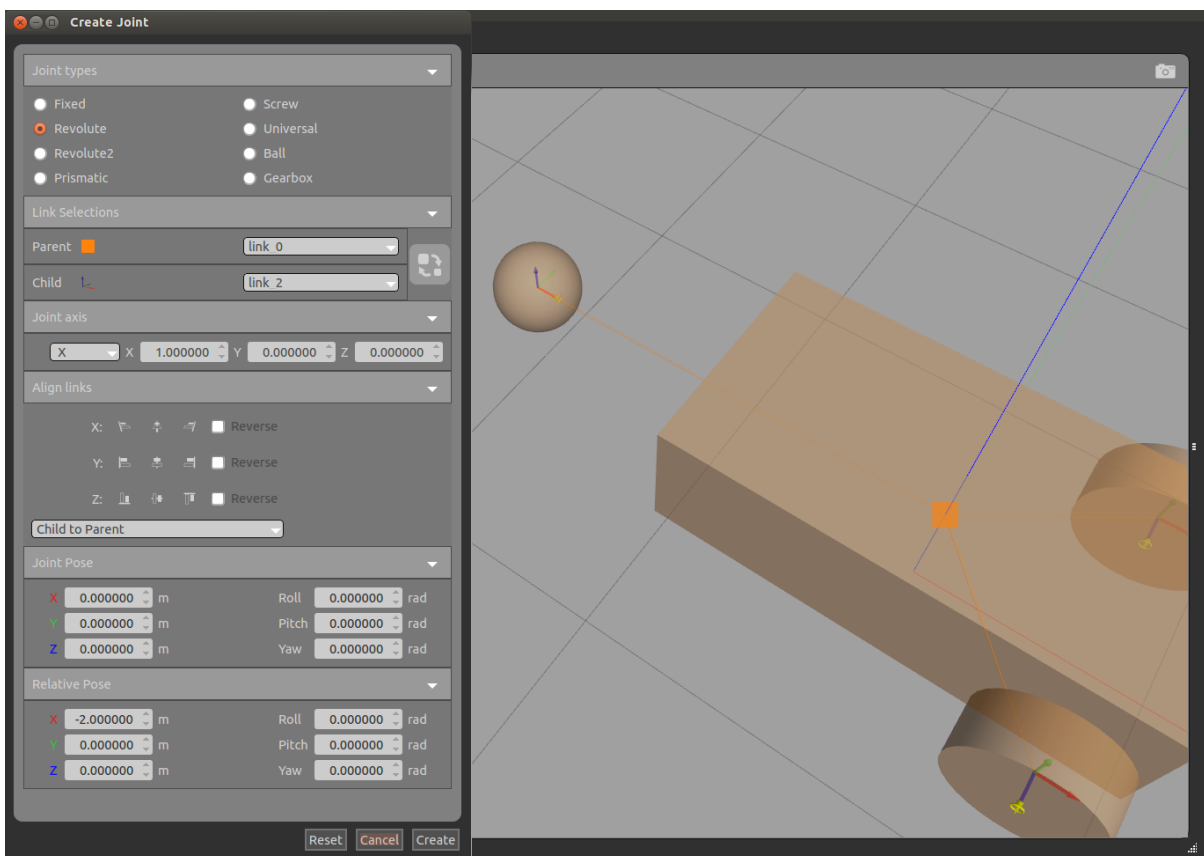
1. To make a caster wheel for the vehicle, click on the Sphere button on the Left Panel and insert it into the scene.



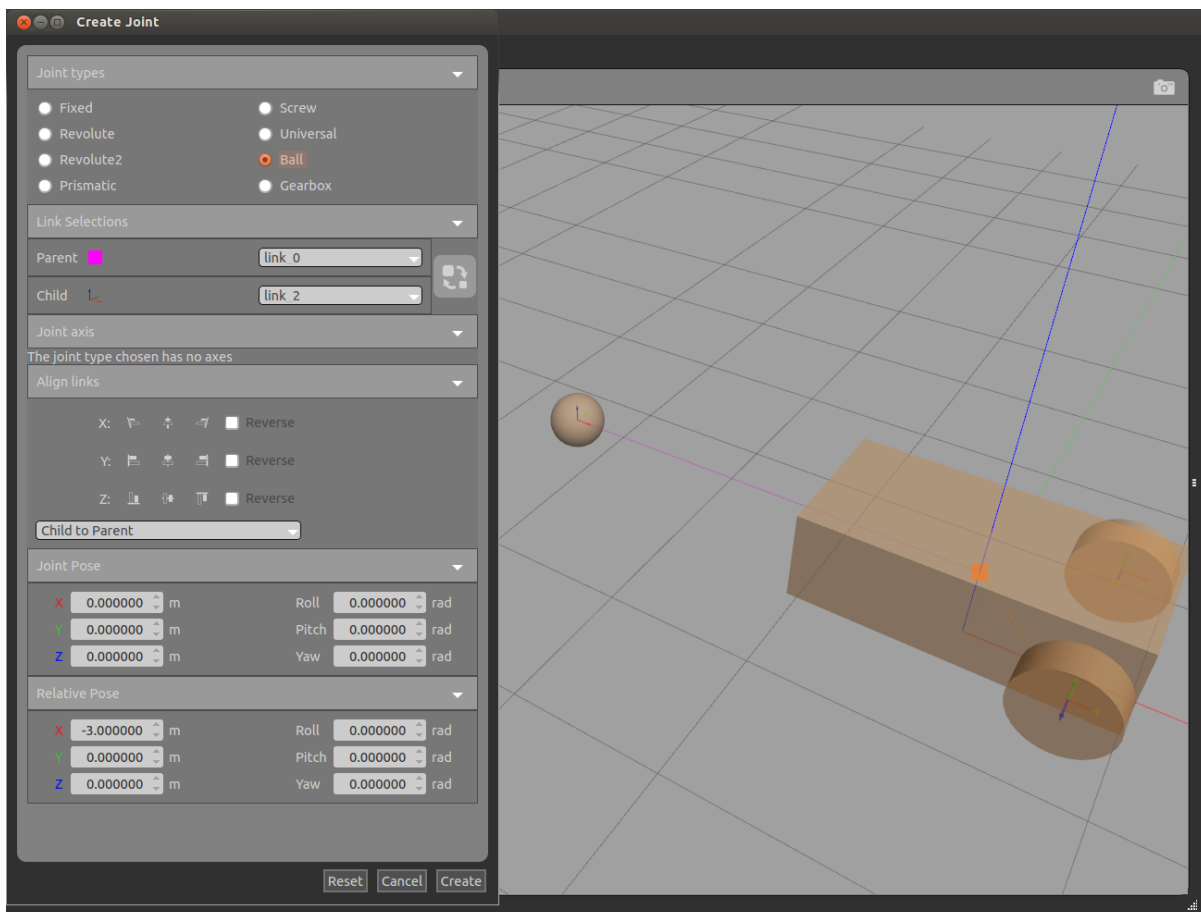
2. Resize the sphere by giving it exact dimensions in the same way you did for the front wheels. Go to the Visual tab to see the list of visuals in this link, expand the only visual item, scroll down to the **Geometry** section and change the **Radius** to 0.2m. Make sure to also do the same to the collision in the Collision tab.



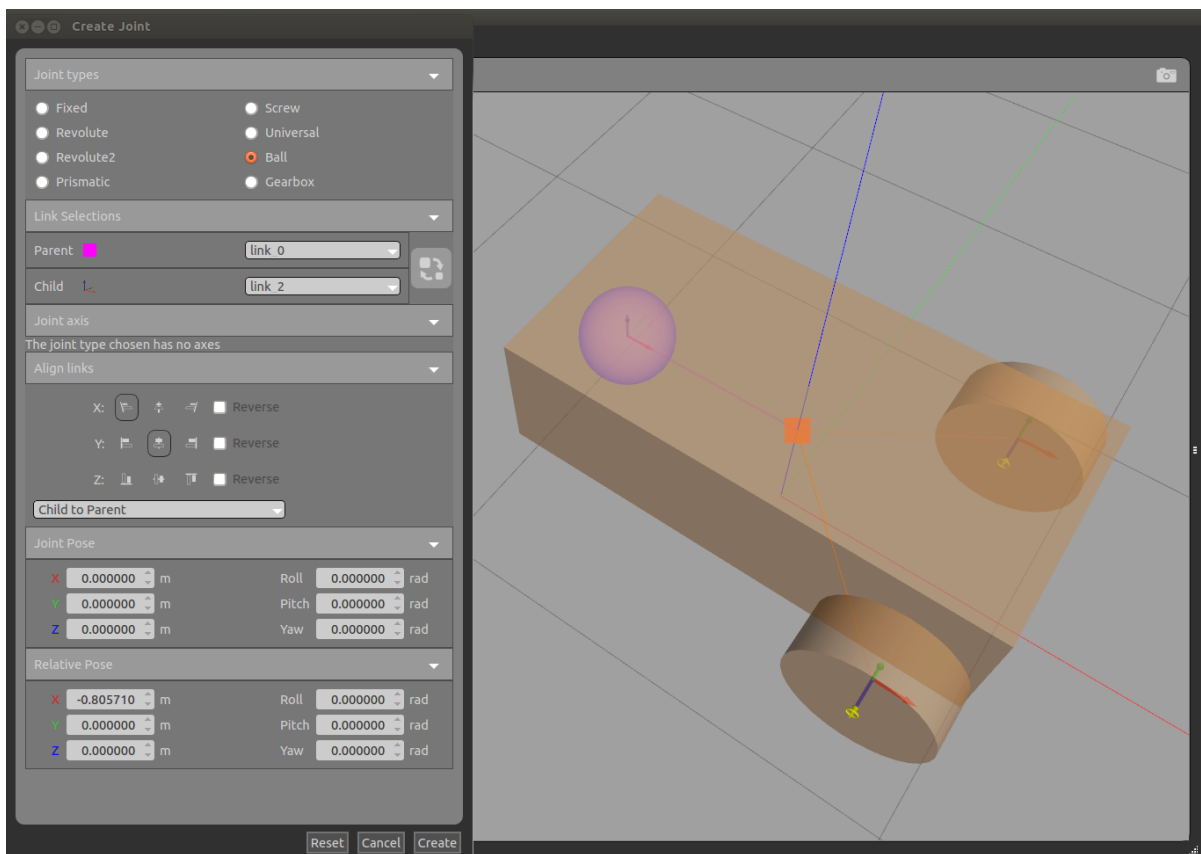
3. To create a joint between the caster wheel and the chassis, bring up the Joint Creation dialog by clicking on the Joint icon in the top Toolbar. Move the mouse to the Scene and select the chassis as the parent link and the sphere as the child link.



4. Unlike the front wheel joints, a caster wheel rolls in all directions and does not have a specific axis of rotation. In Gazebo, this is simulated using a ball joint. So under the **Joint** types section, select the **Ball** joint option. You should see the joint visual in the scene change color to indicate a different joint type has been set.



5. Next, align the caster wheel so that it is centered with the chassis and positioned at the rear end. In the Align links section, select the **Y Align Center** option to center the two links in the **Y** axis, and select the **x Align Min** option to move the caster wheel so it is placed right at the back of the vehicle. Press the **create** button to finish the joint creation process.

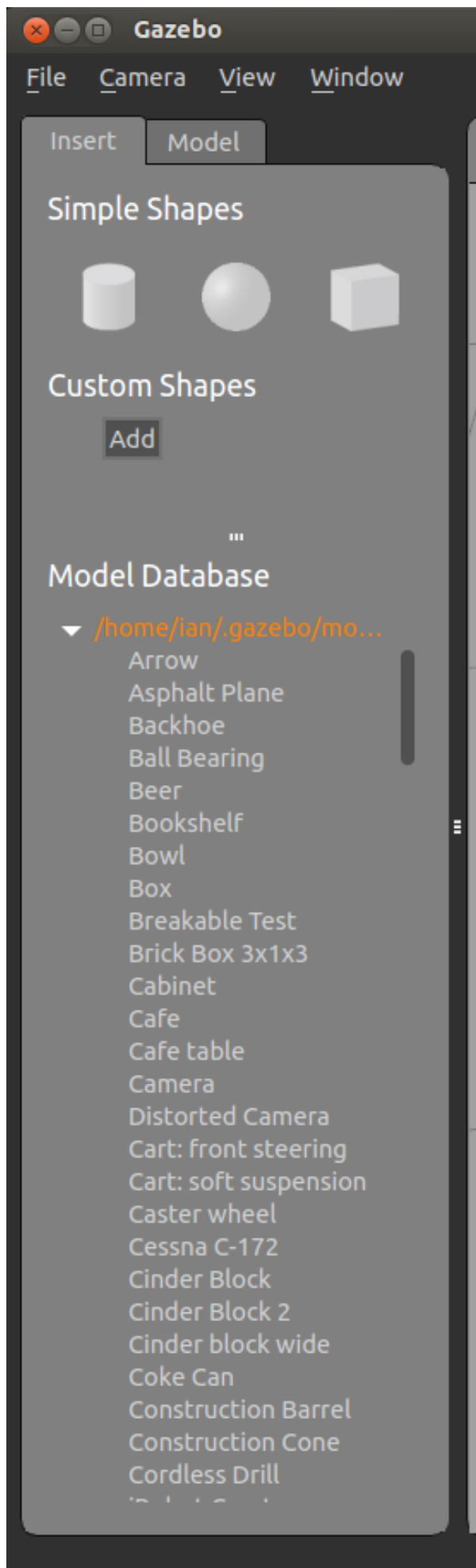


6. Finally, position the caster wheel so that it sits just above the ground. Do this by opening the Link Inspector and setting the `z` position to 0.2m.

Adding a sensor

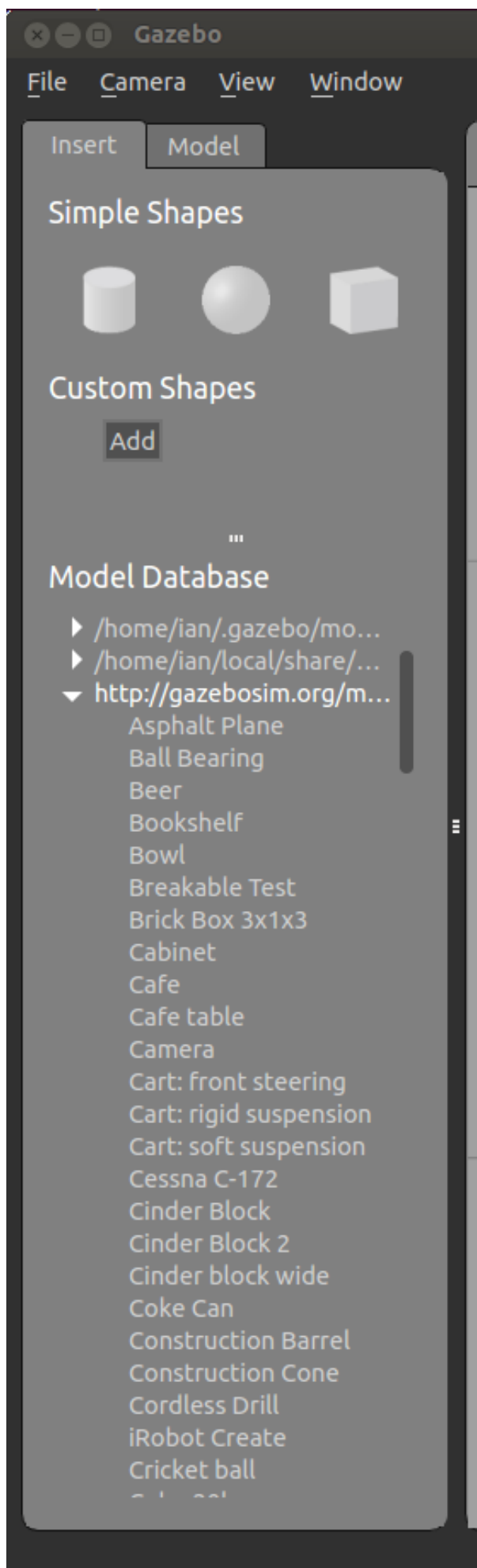
The sensor we will add to the car is a depth camera sensor which is going to help us detect objects in front of the car. In this tutorial, we will insert an existing sensor model from the model database.

1. Go to the Palette (left panel) and select the **Insert** tab to see a list of models available in the **Model Database** section.



2. The models in the lists are organized by the path in which they are located. As you can

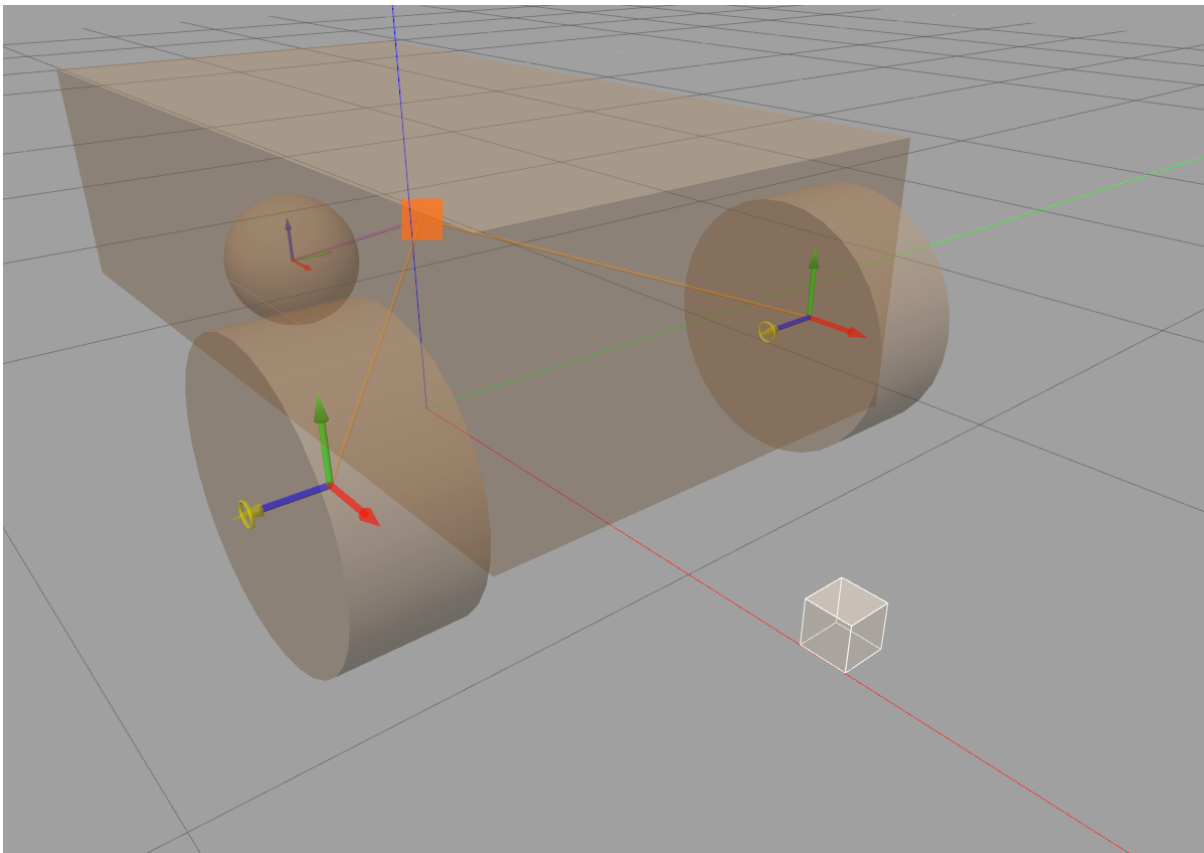
see, the first list contains models available on your local machine as indicated by the path in the title. If you are a first-time user, you may not see many models in the lists. More will appear as you download them from the online model database. Find the list with the path <http://gazebo.org/models/> and expand it to see models available from the online model database.



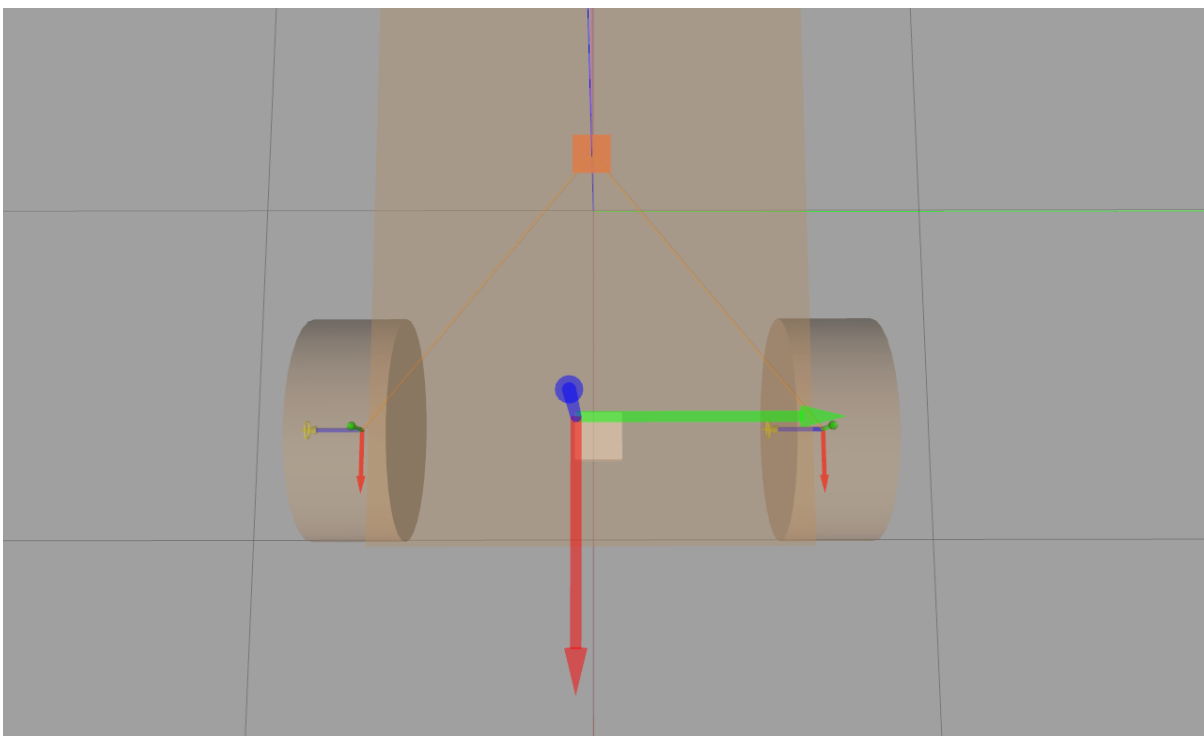
3. The models are arranged in alphabetical order. Find Depth Camera in the list and click on

it to start downloading the model. This may take a few seconds depending on the network connectivity.

4. Once the download is complete, you should see the depth camera model appear in the Scene. It looks like a small cube. Move the mouse over to the Scene and click on an empty space in front of the car to insert the depth camera.

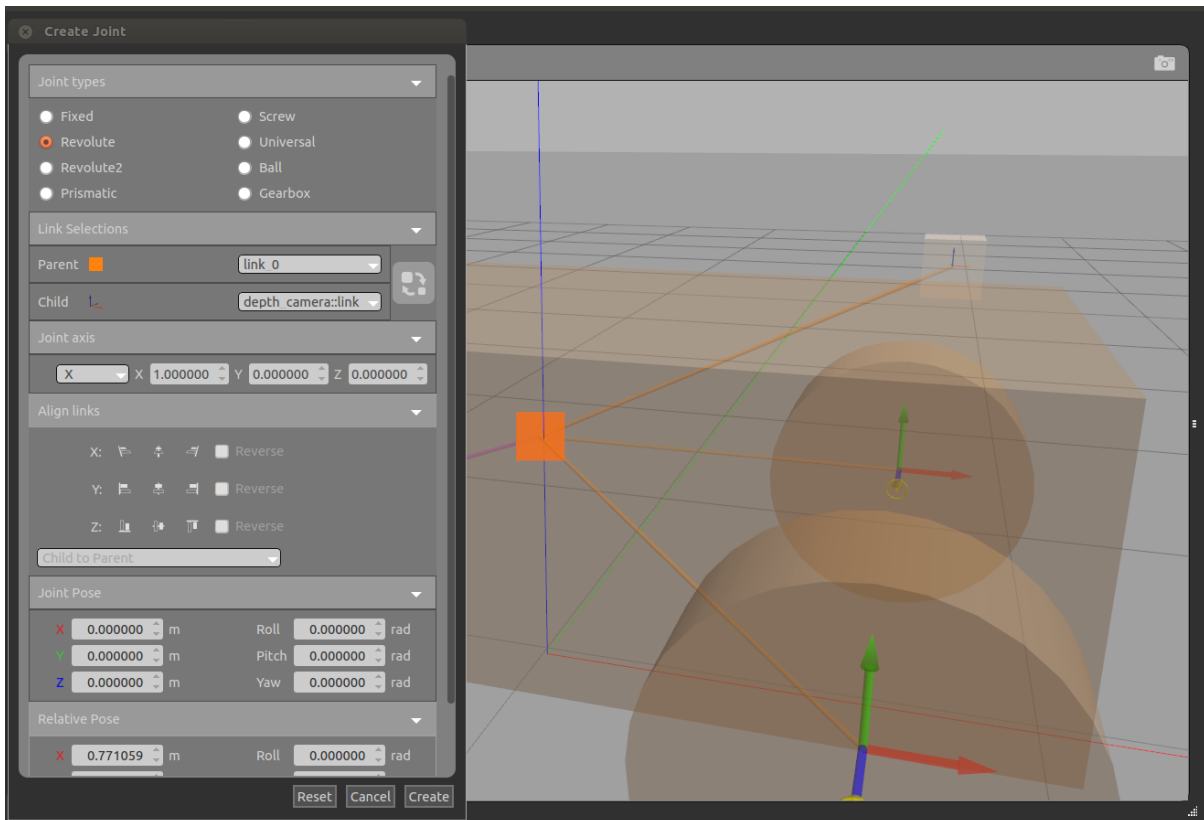


5. Select the Translate tool in the top Toolbar and move the depth camera so that it sits on top of the chassis at the front of the vehicle and roughly centered in the y axis.

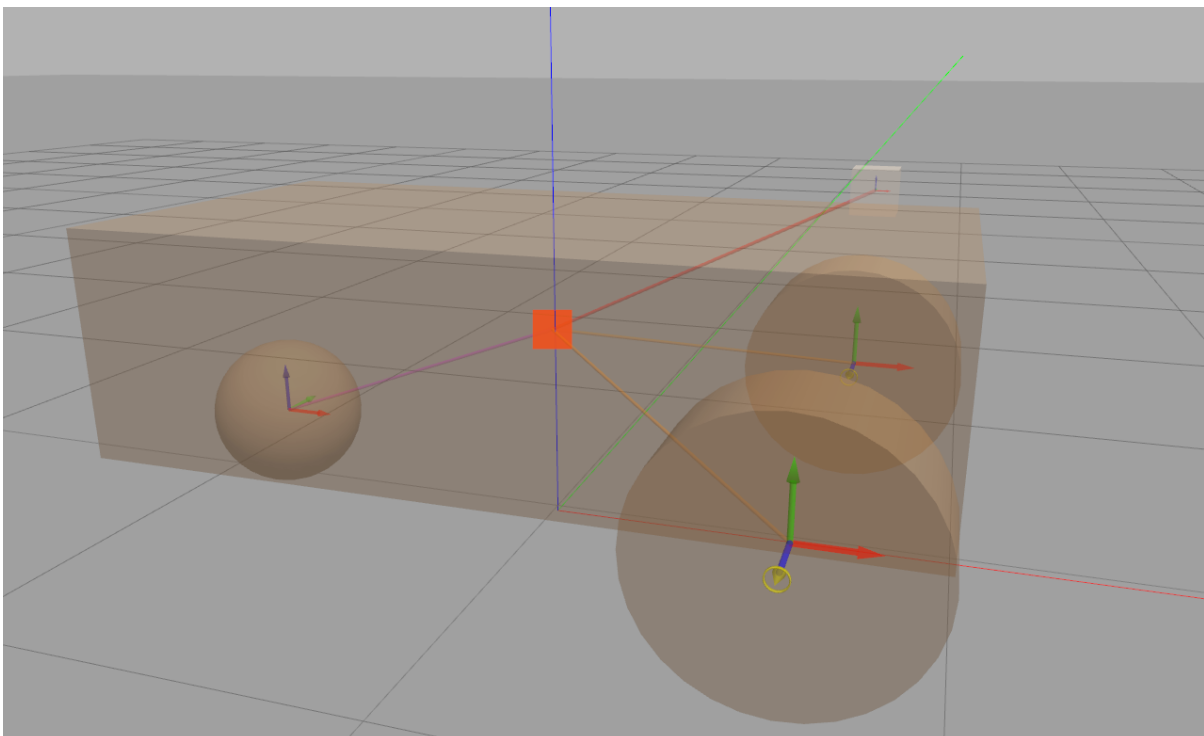


6. Next, fix the depth camera to the chassis. Click on the Joint icon in the top Toolbar to open the Joint Creation dialog. Move the mouse to the Scene and select the chassis as

the parent link and the depth camera as the child link.



7. In the Joint Creation dialog under the **Joint Types** section, select the **Fixed** joint option, and click on **create** to finish creating the joint.

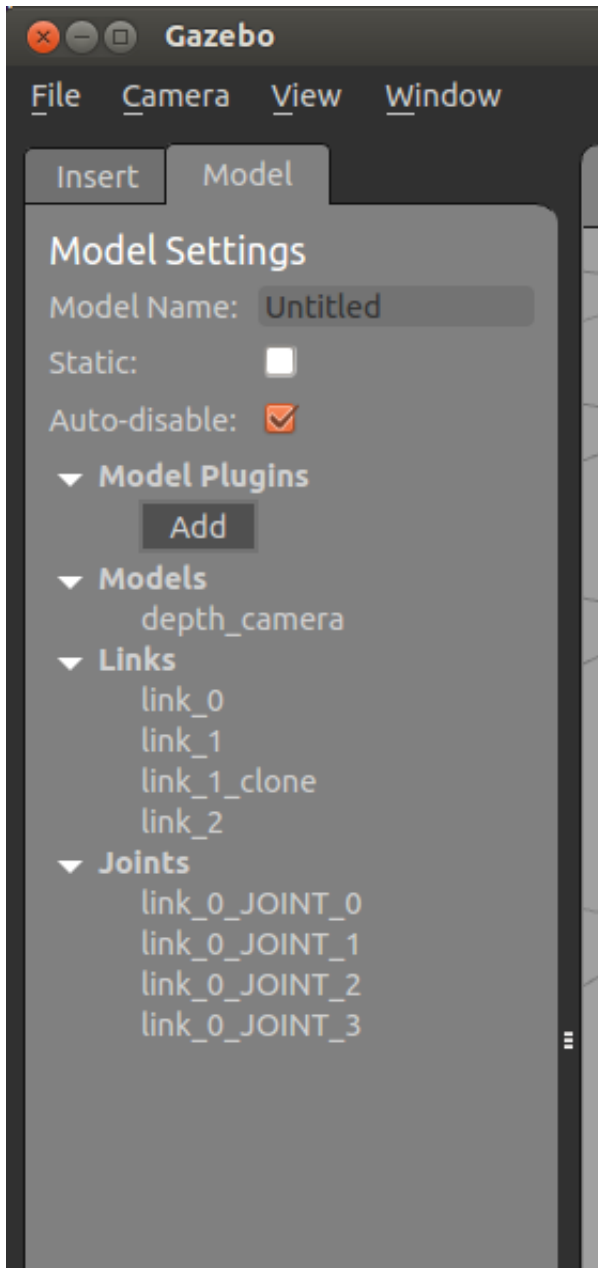


Adding a plugin

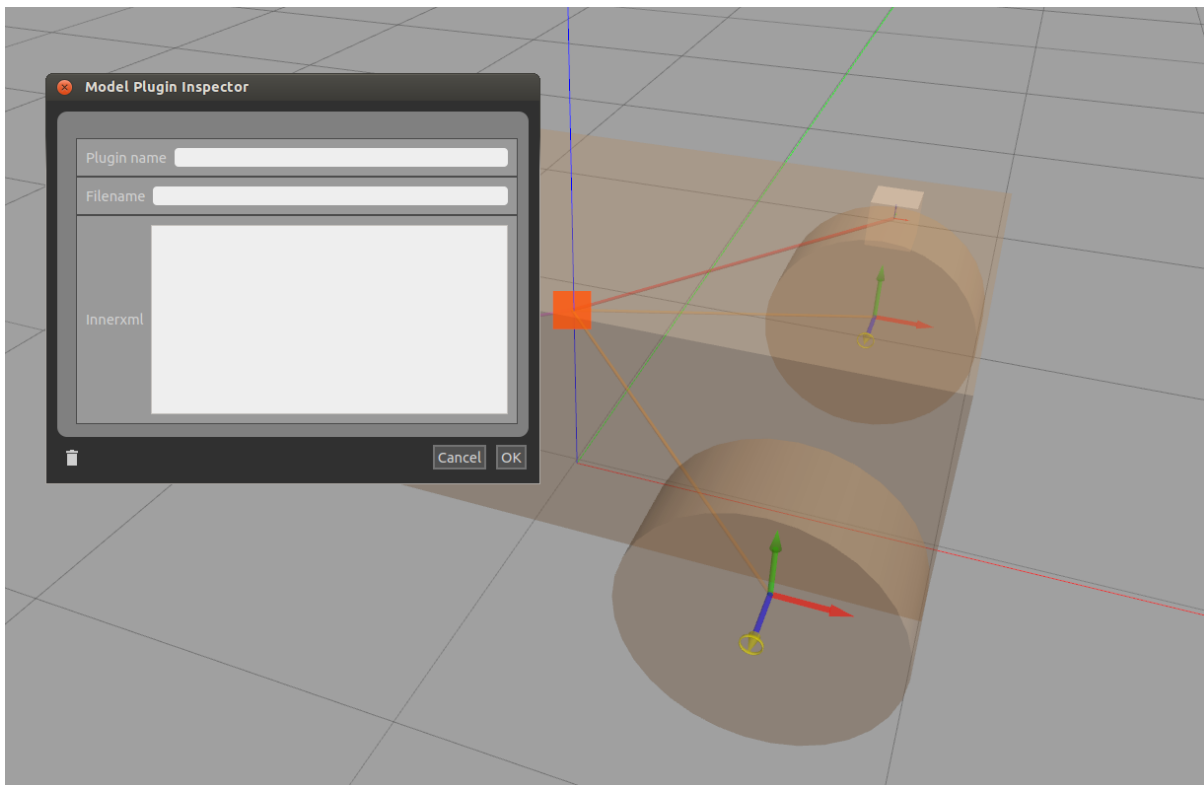
The vehicle we have built so far is complete with all of the physical and sensor components. However, it will not really do much but stay still and generate depth data in simulation. Plugins are a great way to enhance the model with some autonomy by allowing it to perform computations such as sensor data processing, path planning, and control. For simplicity, this tutorial will use an existing plugin for our vehicle. Note that it is possible to create your own

plugins but it requires writing code. See the Plugin tutorials (http://gazebosim.org/tutorials?cat=write_plugin).

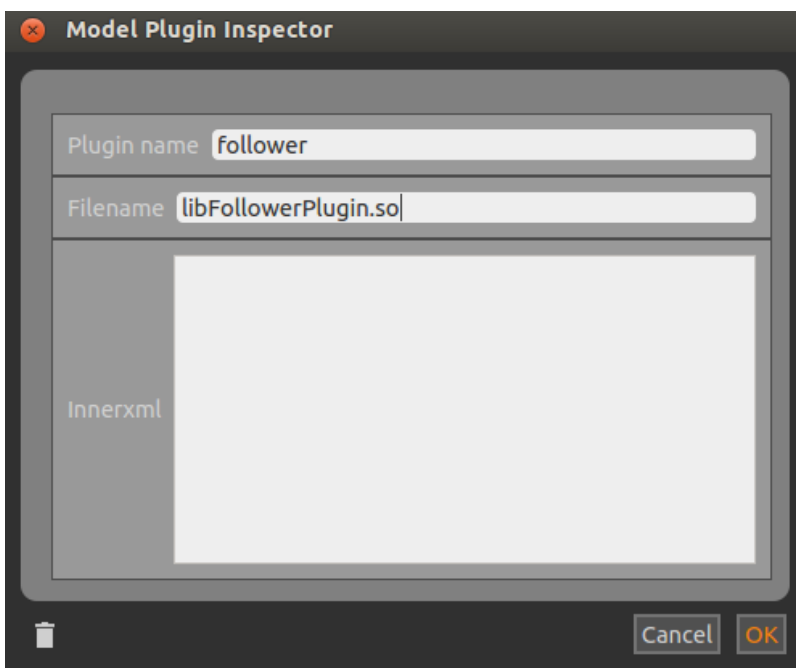
1. Go to the left panel and select the **Model1** tab to see the parts that make up the car model you built.



2. Under **Model Plugins**, you should see an **Add** button. Click on it to bring up a Model Plugin Inspector that lets you add a new plugin to the model.



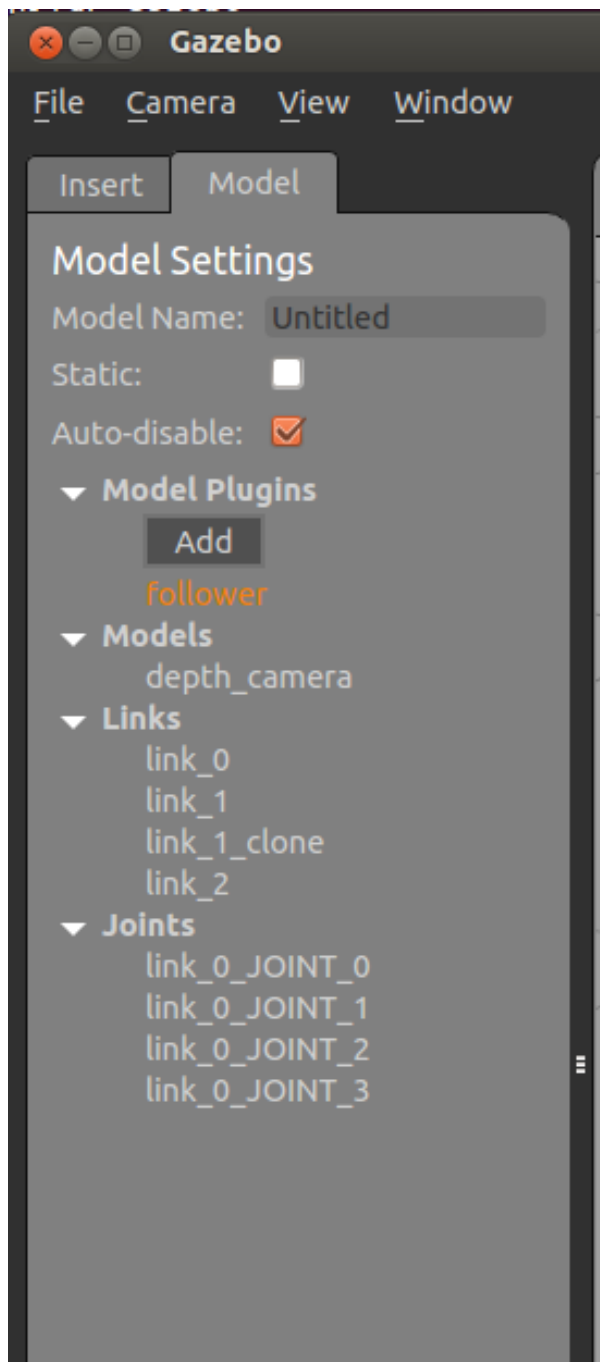
3. First, give the plugin a name. Enter `follower` in the **Plugin Name** field. The plugin name has to be unique within this model.
4. The plugin we are going to use is called `libFollowerPlugin.so` so enter this in the **Filename** field. The filename corresponds to the actual filename of the plugin library stored on your local machine. It exists in the form of a dynamically linked shared object library, hence the naming convention and the extension `.so` (on Linux). Do not worry if you are using Gazebo on other operating systems as the extension will be automatically replaced with the correct one.



5. The `follower` plugin does not require any additional parameters so you can leave the **Innerxml** field empty. Note: This is a simple plugin for demonstration purposes. Plugins typically have various parameters associated, e.g. a differential drive plugin requires specifying the name of joints controlling the left and right wheels so it can move the

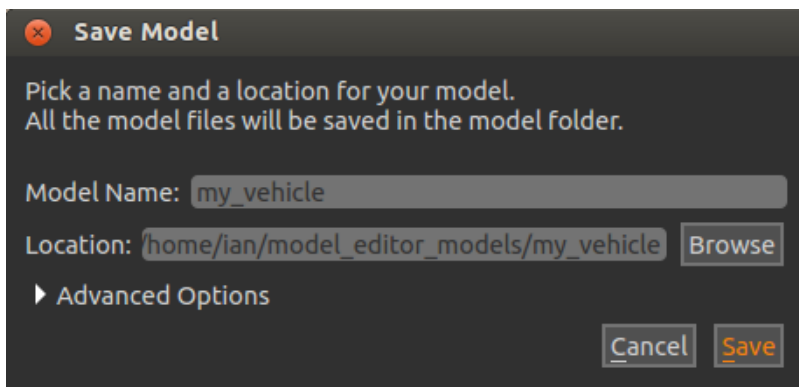
vehicle in the correct direction. In the case of the `follower` plugin, it makes many assumptions on the type of model it is attached to and tries to find the joints and sensor automatically.

6. Click `ok` to add the plugin. The plugin should now appear under **Model Plugins** in the left panel.

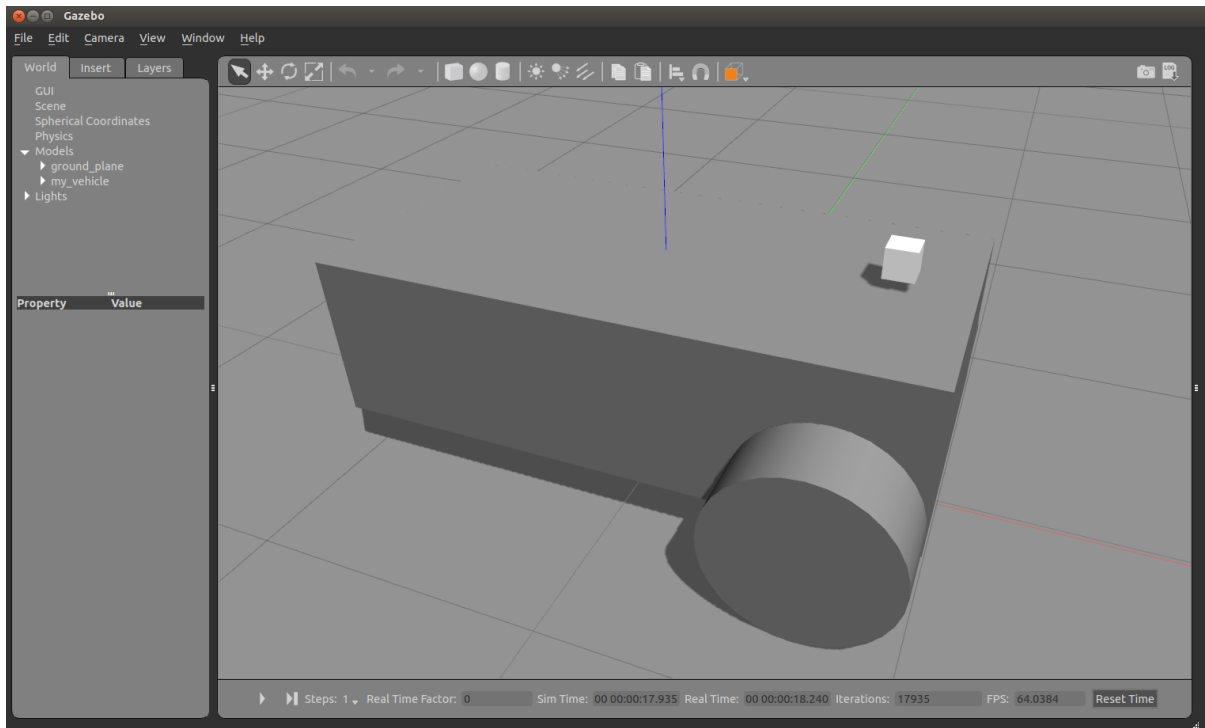


Save your model

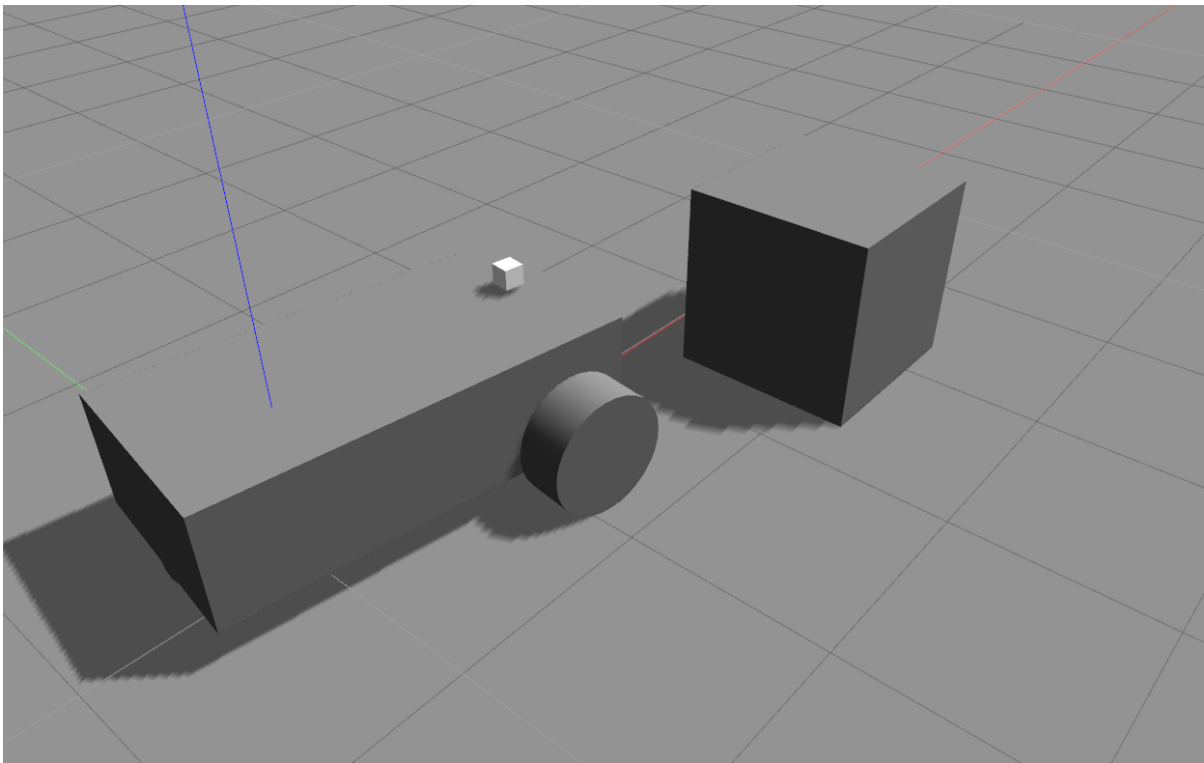
1. Save the model by going to the **File** menu and selecting **Save As** . Enter a name for the model and click **Save** .



2. Exit the Model Editor by going to **File** and selecting **Exit Model Editor** . Gazebo should now switch back to normal simulation mode. Hit the Play button to run the simulation.





3. To test that the plugin is working, insert a box in front of the car and see the car move slowly towards it.



If you want to edit the model again later, just right-click on it and select **Edit Model** in the context menu.

©2014 Open Source
Robotics Foundation

Gazebo is open-source
licensed under Apache 2.0
(<http://www.apache.org/licenses/LICENSE-2.0.html>)

 ([//plus.google.com/u/0/115981436296571800301](https://plus.google.com/u/0/115981436296571800301))
 (<https://www.youtube.com>)