The Unofficial guide for those who are COMPLETELY Lost...

Intro to SketchyPhysics

What is SketchyPhysics and How Can I Get It?

Sketchyphysics is a plugin written by C. Phillips for Google's 3-D program Sketchup. It allows the objects you draw in Sketchup to "come to life" in a real world simulation with gravity, collisions and interactions with other objects.

If you want to install it, you must first have the latest version of Sketchup installed. (It is free.) Then do download SketchyPhysics and install it.

It is an amazing addition to Sketchup, but finding any help, even a simple starters guide is next to impossible.

If you've seen the You-Tube video of even the simple car rolling down a hill and then tried to make it, you'll see exactly how frustrating SP can be when wheels float in the air, the body falls off, or the damn thing just won't move!

My goal of these few humble pages is to provide a starting point so anyone who knows Sketchup can start experimenting with the physics.

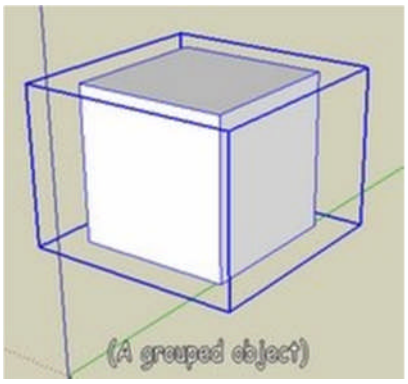
This isn't going to be a comprehensive help site, but rather a diving board so you can start the learning process on your own. This also won't be a Sketchup guide. I assume you are already proficient in Sketchup before starting here.

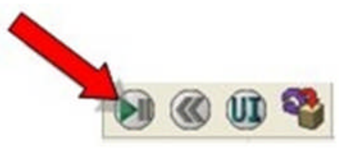
If you need more information than you can find in this guide, stop on by the SketchyPhysics Forum that is a part of the SketchUp Community Forums.

The Basics...

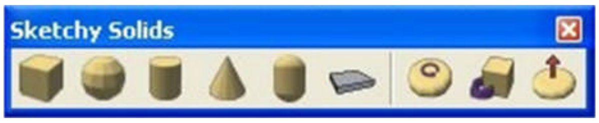
The first thing to note is that SketchyPhysics ONLY WORKS WITH GROUPED OBJECTS.

If you start by making a cube and grouping all the parts into one object, the object will be recognized by SP. Physics will then be applied when you press the RUN button.



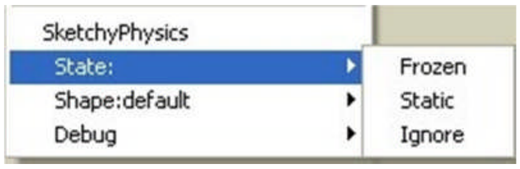


There are also auto-shapes provided through a toolbar that is installed with SketchyPhysics. These shapes are already grouped and ready to use, all you have to do is draw them where you want to in your model.



Setting the STATE of the Object

After making a grouped object, there are 2 main attributes to use when creating an object to use in SP. They are accessed by **Right-Clicking** a grouped object.



The first option is **STATE**. This sets how the object behaves in SP.

The STATE options are:

FROZEN: Image will stay in place (it won't fall) until it is hit by another object, then regular Physics takes over.

STATIC: The object will NEVER move, no matter what happens to it. Other objects will bounce off of it.

IGNORE: The object will not move, and other objects will not interact with it. They will pass through it. (It is a ghost.)

On the next page we'll discuss the second option: **Shape**.

***Continue on to SHAPES*.**

Shape Physics

The other two right-menu options are SHAPE and DEBUG, and they are used together.

SHAPE: Sets the shape that SP sees the group as.

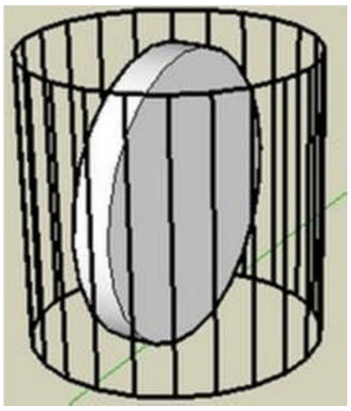
Just because you grouped a Cube doesn't mean SP "sees" it as a Cube!

USE THE SHAPE OPTION ON EVERY OBJECT YOU CREATE FOR SP!!!!

DEBUG: Use this option to check the shape SP "sees". After grouping an object and setting its SP shape, select this option and press RUN. SP will make a black wire diagram of the object AS IT SEES IT.

You may be surprised at what it sees. Also, just because you make a cylinder and set the shape to cylinder, doesn't mean it will be oriented correctly. (See below.)

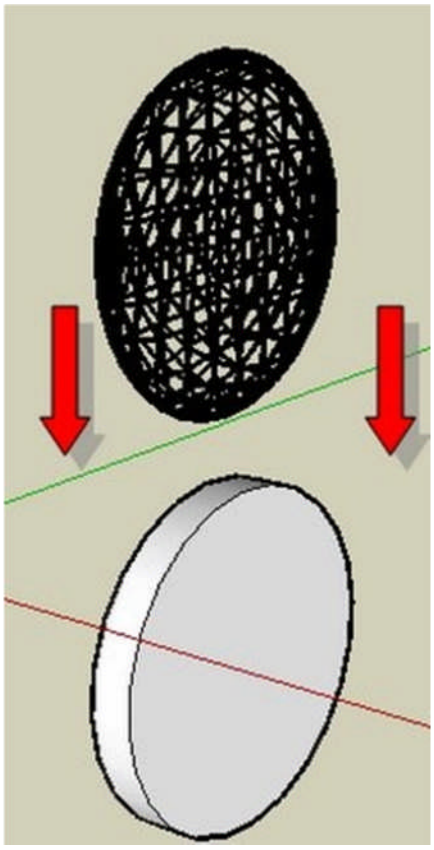
AN INCORRECTLY SHAPED OBJECT:



This cylinder object was set to "cylinder", but by using the DEBUG option, you can see the black wireframe shows SP thinks it is actually on it's side. If any other object interacts with this cylinder, it will hit the black wireframe, NOT the actual grouped shape!

You can purposely set objects to the wrong shape for fun. For example, if I turned this cylinder on its side, it would "roll" end over end like a quarter being flipped.

So what can you do if the shape gets set wrong? Try another shape. "SPHERE" actually works quite well for a short cylinder. SP will squash a sphere to fit the cylinder shape relatively well. (You can see the original wireframe as the cylinder falls below.)

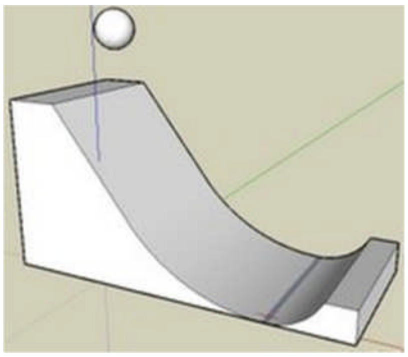


Advanced shape physics

There are a few more shapes that are provided for you when the normal ones don't work out.

The first one to note is "STATIC MESH". This draws a mesh around your shape to create the outlines. This is beneficial when making a "hill" for something to roll down, or a multi-pitched slope. However, it is STATIC and UN-MOVEABLE. Therefore this option is mostly used when making a "ground plane" for everything else to "ride" on.

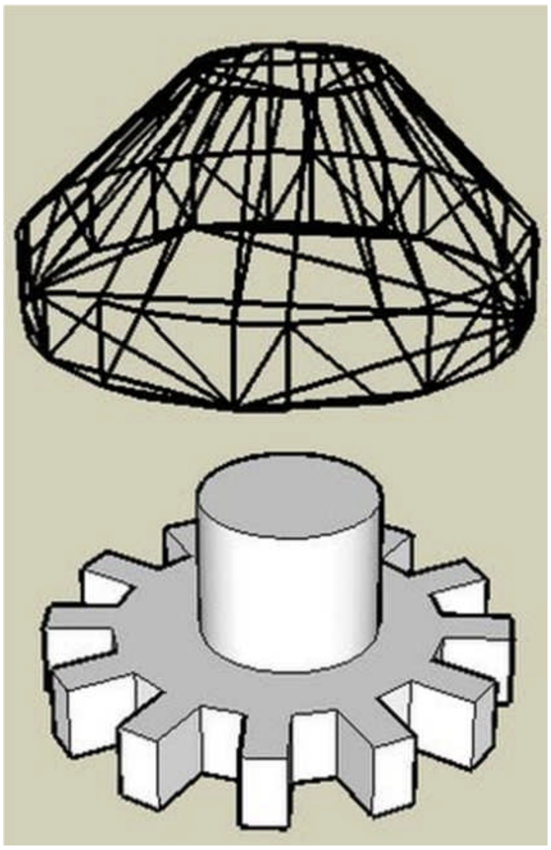
This ramp was made as a "STATIC MESH".



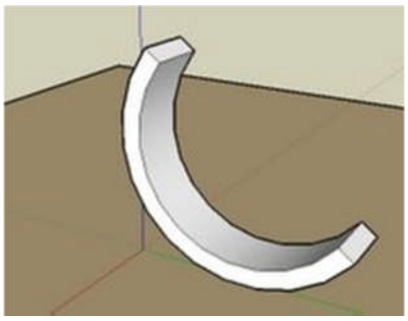
It does not move, however the ball will fall and roll down its slope.

The next shape to note is "CONVEX HULL". And it is exactly what the name describes. It takes a shape from end point to end point and makes a closed "hull". (Like a boat.) This has some limitations, as I'll describe next.

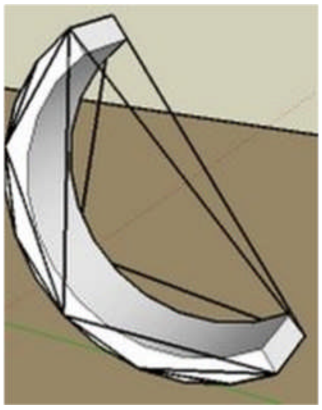
Example of a convex hull:



You can see that the convex hull made a shape similar to the gear above, but if you tried to make it "work" with another gear, it wouldn't. So how do you fix it? Let's look at a similar shape that I had a problem with. I encountered this situation when using "CONVEX HULL" to make an upside-down arch:



The arch wobbled back and forth as you would expect, but when I dropped a ball on it from above, the ball did not fall into the arch. It stopped above it. WTF?!! After using the DEBUG command, I saw this is what was happening:

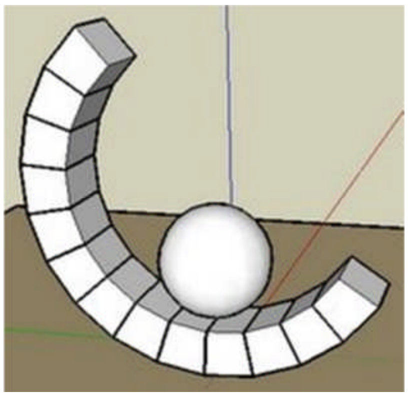


You can see that SP "drew" in some lines to make this a "CONVEX HULL". (No support for a CONCAVE hull as of yet.) And it behaved more like a half circle than an arch. So how to fix it? Hmmm. No standard shape will overcome this. So we move onto...

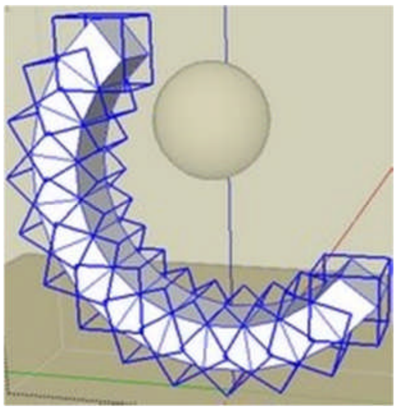
Grouping groups

Even more advanced shape physics.

The solution: Instead of one arch, I built the arch out of a number of groups, (like bricks in a real building) and grouped each sub-group together to make one arch shape.

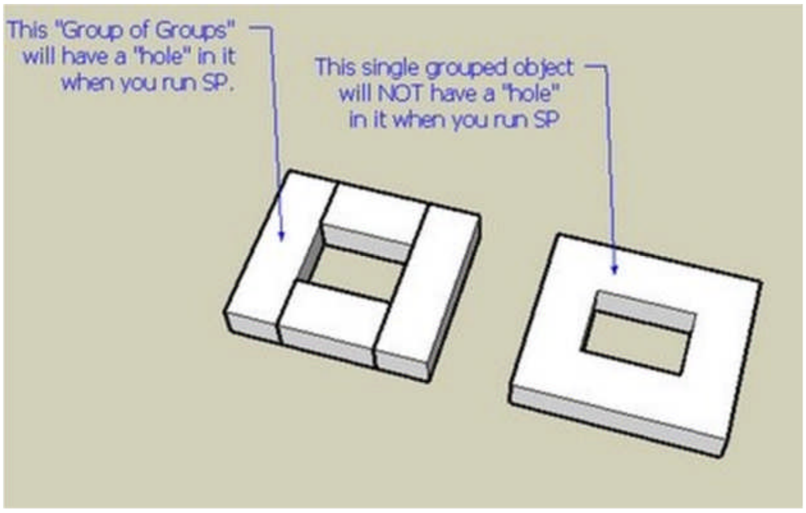


Now the ball falls into the arch and the arch wobbles back and forth on its axis. To better explain this look at the following picture that shows the "group of groups".



Each of these sub-groups are all part of one large group. They do not separate, and they behave as one large object.

Grouping groups is also the only way that I know to make a hole in a moveable object.


This "Group of Groups" will have a "hole" in it when you run SP.
This single grouped object will NOT have a "hole" in it when you run SP

NOTE: You can make a hole in an object using the Static Mesh shape, but then your object cannot move.

Next we'll get into using joints, and work with more than just gravity.

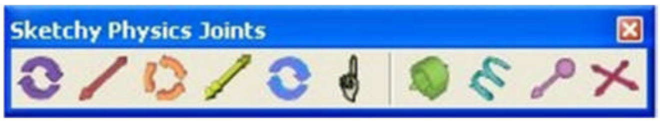
Continue on to Jointing Basics

Jointing Basics

Up until now, we have only made objects that will be affected by gravity and collisions in SP. Now we'll learn how to use joints to make automated or specific movement in our models.

A joint is a special type of grouped object that can "control" other objects in SP.

There are a number of different joints you can choose to add to your model, and each joint has a specific behavior.



The joints also have their own special tool for defining their connections, the Joint Connector tool:

There are three basic steps when adding a joint to a model:

Add a joint for the part you want to make move.

Make the moveable part and the joint a group.

Joint-connect the joint to a “parent” object in the model.

These 3 steps will work for MOST joints in MOST situations.

Before we get started, let’s be clear on the terminology I’ll be using throughout this guide.

JOINT-CONNECTING an object to its joint means using the Joint-Connector tool.

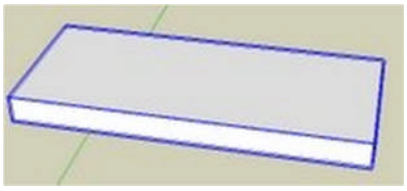
GROUPING an object to its joint means making them a group like any other group in Sketchup.

One of the harder concepts for me to grasp when first working with SP was when to joint-connect an object to a joint, or when to group it with a joint. Thanks to C. Phillips, I now have a better understanding of when to do which.

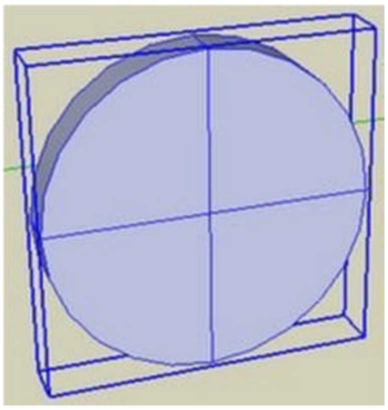
With that being said, let’s move on to an example, and build a car that rolls. If you want to follow along with this tutorial, you can download the basic model HERE.

NOTE: The model just contains the primitive shapes. To make it work, you'll have to follow this tutorial and put it together.

The first thing we'll want to do is make a simple body and group it.

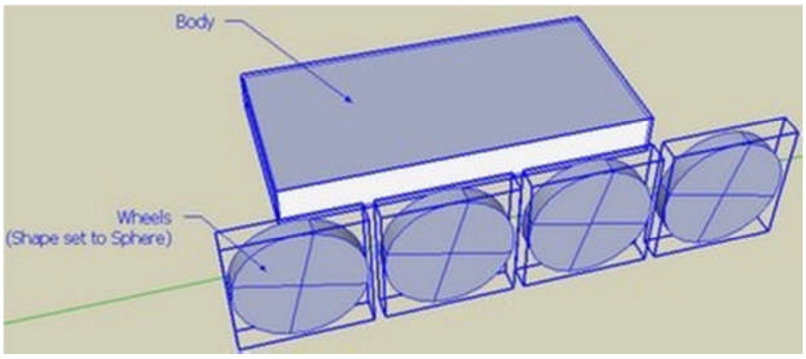


Next we'll make a wheel shape and group it.



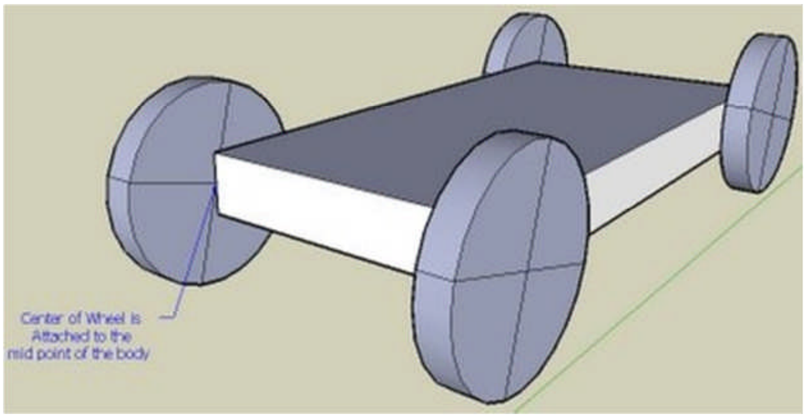
Set the shape of the wheel to "convex-hull" or "sphere" and make 3 copies of it:

(Click any picture below to enlarge)



Now is a good time to Debug the wheels and body to make sure they are shaped properly.

Next move the wheels to where you want them to be on the body. It is easier to move the wheels if they have a design with some sort of center point. (This is why I made them in 4 sections instead of one solid circle.)

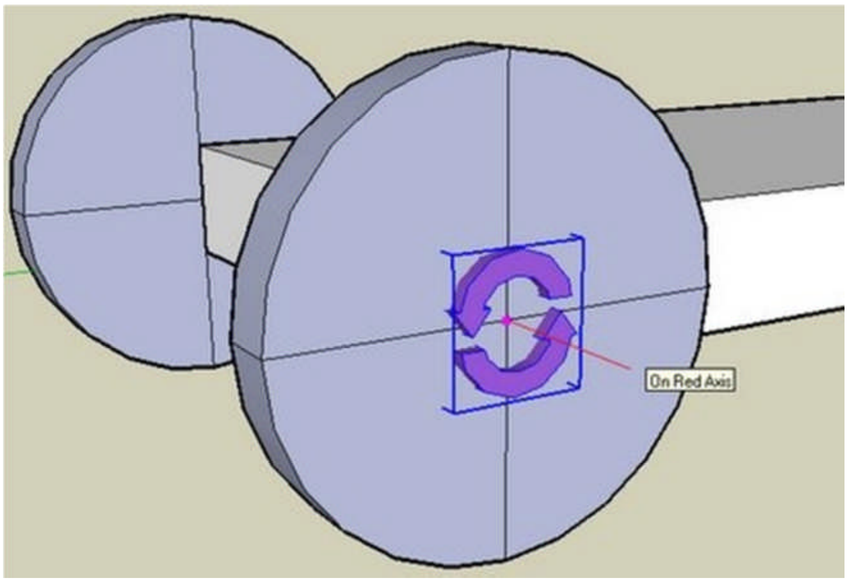


Now let's get to the jointing and grouping and make this thing work.

Remembering the 3 step process mentioned above, we'll start with step one:

I. Add a joint for the part you want to make move.

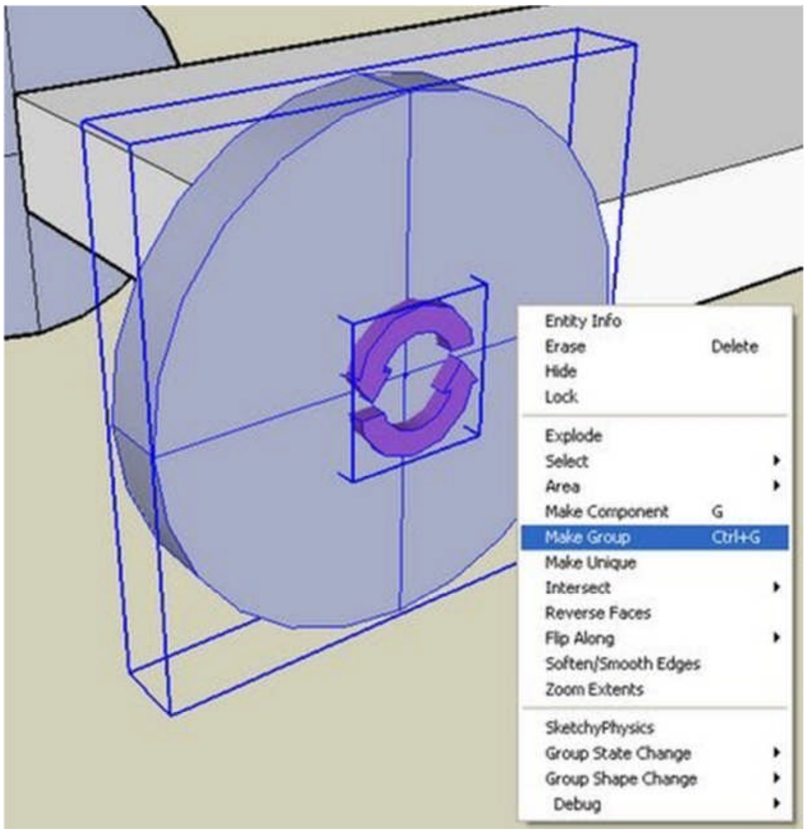
We'll use a **HINGE** joint for each wheel to make them spin. So place a hinge joint at the center of your first wheel and set the axis. This is a two click step. The first one places the joint, and the second click sets the axis depending on where you move your mouse.



Be sure to set the axis correctly!

Now on to step 2:

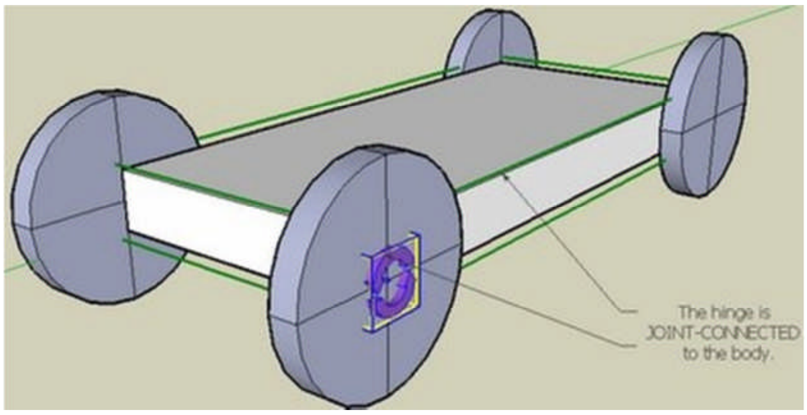
II. Make the moveable part and the joint a group.



By grouping the wheel and the joint, we have made them behave like "one piece". But they still won't spin until we define a "parent" object for them to spin in relation to. So let's move on to step 3:

III. Joint-connect the joint to a “parent” object in the model.

Using the **Joint-Connector** tool, **Joint-Connect** the wheel hinge to the body.



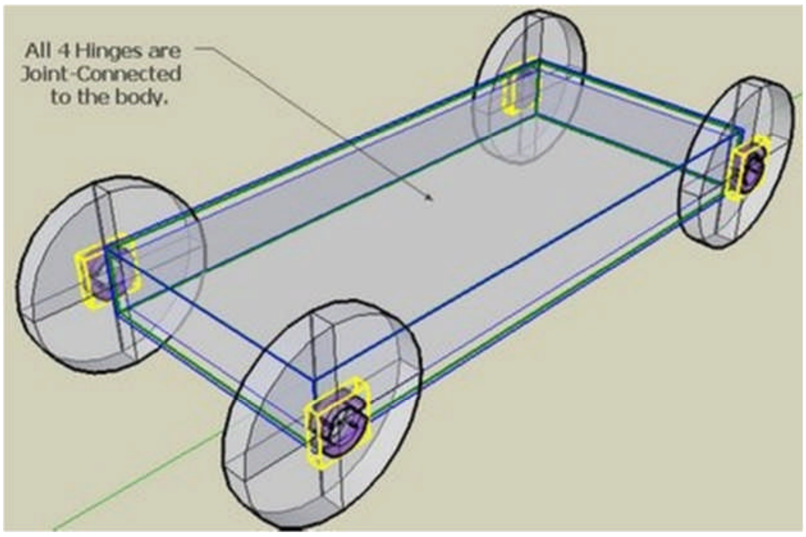
By doing this, we have given the hinge a "parent" object. (The body) It is now connected to the body, and it will spin in relation to the body. And since our wheel is grouped with the hinge, it will spin as the hinge does.

Now repeat the process for the other three wheels. Make a new hinge for each wheel, group it to its respective wheel, and joint connect it to the body.

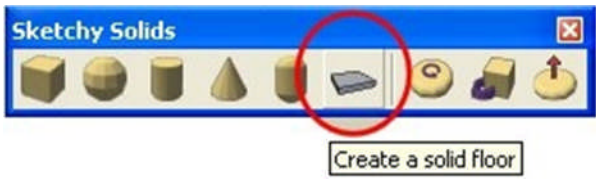
This brings up another point to be made when working with joints. I have found that copying joints leads to unexpected results, so don't copy the hinge for the other three wheels. Be sure to make a new hinge for each wheel.

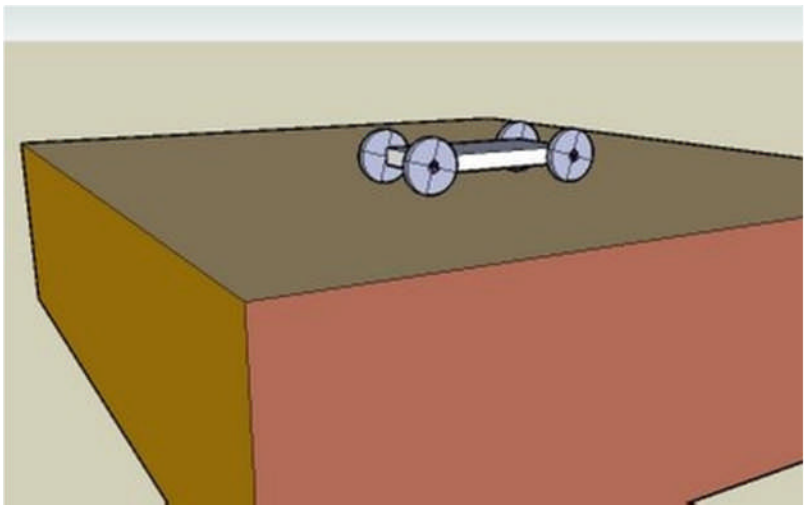
So As a rule of thumb: Don't copy joints.

In the end, you should have all four hinges joint-connected to the body, making it a parent to all the hinges. And since the wheels are grouped to their hinges, the entire model will behave as one big family.



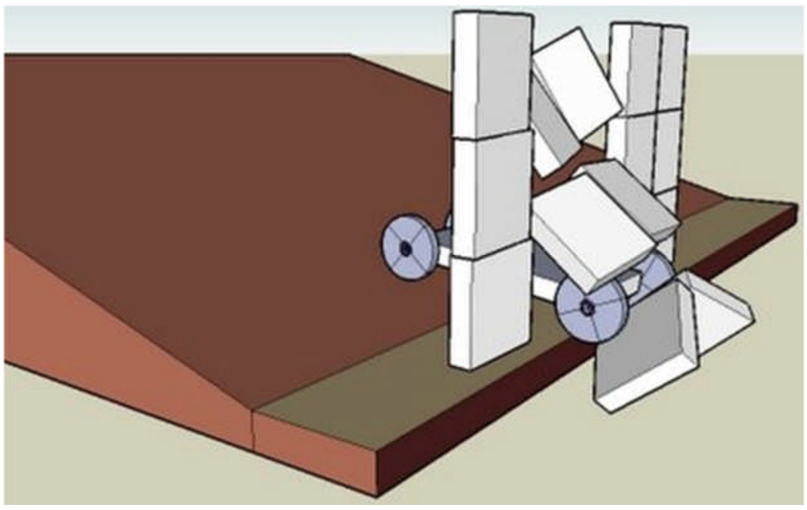
It's now time to test our car. If we RUN the animation now, the entire car would fall into space as one object. So let's make a ground plane for the car to rest on. You can either make a ground plane yourself and set it to "STATIC MESH" or you can use the auto-shape (SOLID FLOOR) to make the plane in one click.





RUN the animation and you should be able to drag the car around the ground plane with the wheels spinning as it moves.

From this point there are a number of options to make the car move on its own. You could make one of the joints a motor instead of a hinge. Or you could slope the ground plane to watch it roll down hill.



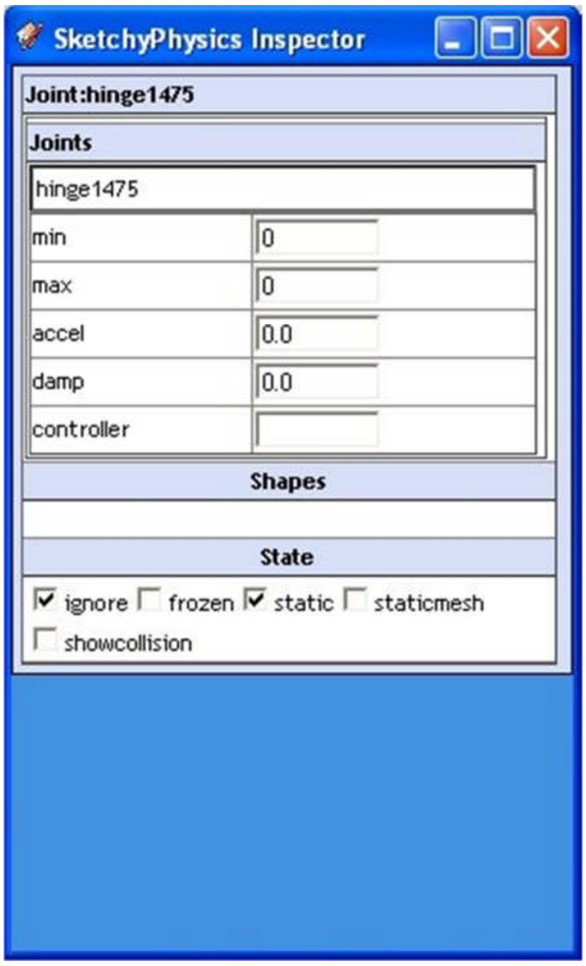
If you want to get even more advanced and add forward, reverse and steering to your car: Continue on to the next two pages where we'll learn about using the UI.

Continu on to User Interface

The Inspector window

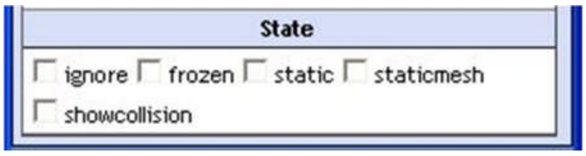
Now that we have the basics of using SP down, let's move on to some of the more advanced features available.

One of the first things you notice after installing SP is the **SketchyPhysics Inspector** window (otherwise known as the **UI**) that pops up whenever you start SU.



The UI contains information and adjustable settings related to the object selected in your model. Anytime you select an object in your model, this window shows the general properties of that object. This is particularly useful when checking joint properties.

Let's take a look at the bottom of the UI first. You'll see there are a series of check-boxes under the **STATE** header:



You can set the state of the object here just like you do by right-clicking the object.

One important difference is the **showcollision** check box. This is similiar to the "debug" command when right-clicking, except instead of drawing a wire-mesh only in the initial position of the object, the wire mesh is visible throughout the objects movement. Now you can see when and how objects collide, not just how they look when they start.

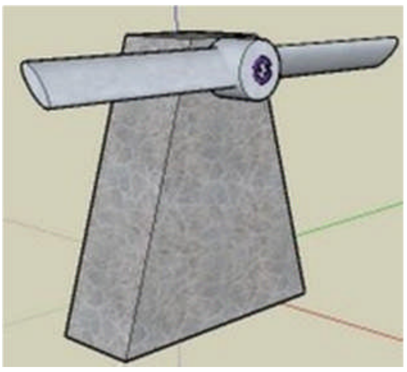
Further up in the UI, you will see numerous settings for joints such as: **min**, **max**, **accel**, **damp** and **controller**. Select a joint first, and then use these settings to add further control to your selected joint.

Note that these options depend on what type of a joint is selected. Some, all, or none of these options may be available depending on the joint type.

To see how powerful the UI can be, lets start with a simple Hinge joint and see what kind of modifications we can do to it through the UI. **Through the UI, we can make a simple hinge behave like a servo, or even a customized motor with forward and reverse capabilities!**

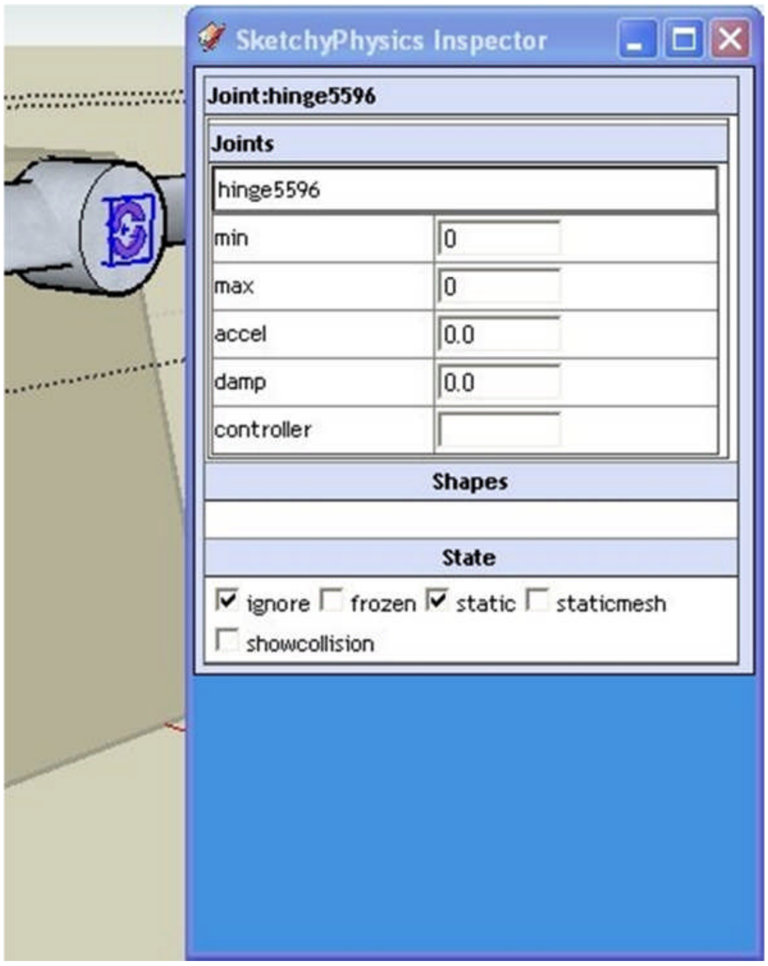
So lets begin by creating a simple model with a hinge in it for an example. Below is a windmill with a single hinge that we'll use as the main controllable object.

(Download the model HERE and follow along.)



(The hinge is grouped to the blade and joint-connected to the base.)

Now select the hinge and look at the UI window.



SketchyPhysics Inspector

Joint:hinge5596

Joints

hinge5596

min	0
max	0
accel	0.0
damp	0.0
controller	

Shapes

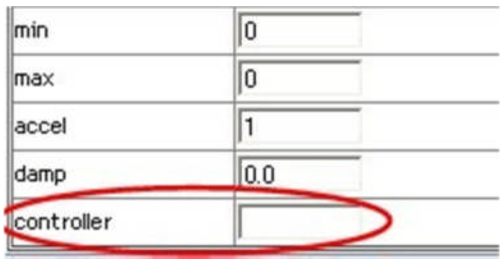
State

[✓] ignore [ ] frozen [✓] static [ ] staticmesh
[ ] showcollision

The first settings you see are the **Min** and **Max** settings. These can set the range of movement for the Hinge. In a normal hinge, these settings are both at zero, and so the hinge spins in a 360 degree circle, like on a wheel.

If we change the min and max to -90 and 90, this will limit the swing to 90 degrees in either direction of the start point. (This would be useful in a door where you don't want it to swing past the normal extents.) For now, we'll leave these at the default settings so our windmill can spin 360 degrees.

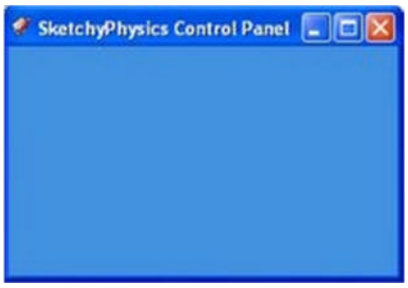
The next thing we'll look at is the "controller" box. This can be looked at as a way to "name" a joint.



min	0
max	0
accel	1
damp	0.0
controller	

For a hinge this is blank by default. The reason it is blank is because normally a hinge does not need any control.

Whenever we run a Sketchyphysics animation, a control panel pops up. This control panel contains all the "named" joints in the animation.

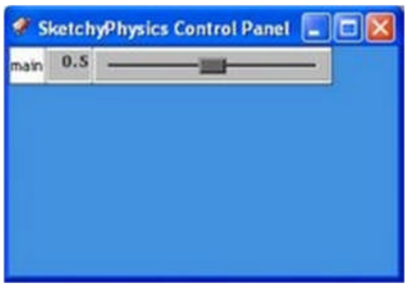


If we ran the animation now, the control panel would be blank. This is because our hinge has no "controller" value yet.

The control panel only shows the joints with a "controller" value set to them.

Once we "name" the hinge, it will show up in the **"control panel"** when we press RUN. So let's name our hinge "MAIN" in the UI and run the animation.

Now the named hinge "main" will now show up in our control panel when we run the animation:



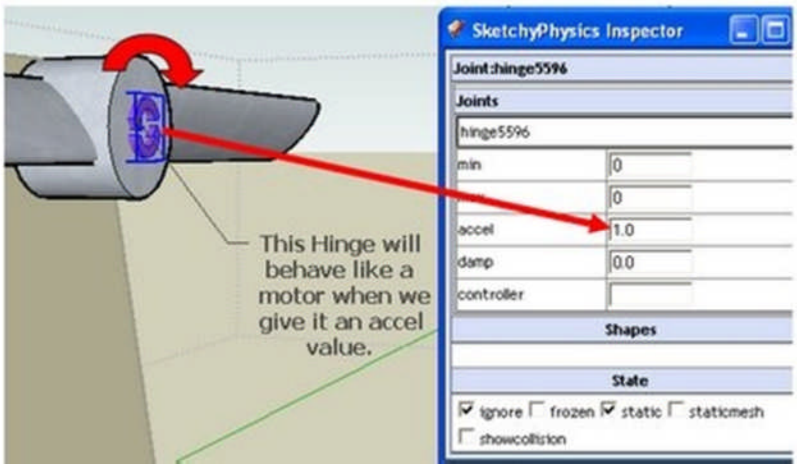
Since we named our hinge, and gave it control through the control panel, it will no longer behave like a hinge spinning freely. Instead it will now be a controllable joint like a servo. As you move the slider left or right, the hinge will move in that direction, but it will no longer spin freely. In essence we have created a servo from a hinge simply by naming it in the UI.

Now lets go back to the other settings in the UI and see what else we can do with this hinge.

Next we'll look at the **accel** option. This relates to the automatic speed of the joints movement. On a hinge this is normally zero, and the hinge does not spin unless something makes it move. If we set an accel value, the hinge will spin on its own and behave more like a motor.

One important difference is that the hinge can spin in either direction, where a motor can only spin in the direction it is originally placed.

Let's set the accel value to 1 and see what happens when we press RUN.


This Hinge will behave like a motor when we give it an accel value.
SketchyPhysics Inspector
Joint:hinge5596
Joints
Shapes
State

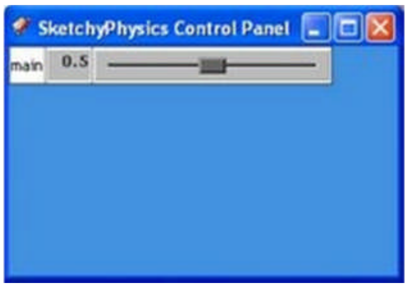
hinge5596

min 0
0
accel 1.0
damp 0.0
controller

✓ ignore ☐ frozen ✓ static ☐ staticmesh
☐ showcollision

If you did not name the hinge by setting a value in the controller box, the hinge will spin in one direction just like a motor.

If you did name the hinge in the "controller" box, you'll see the control panel pop up with the name of the hinge:



main 0.5

By default, the slider bar is at 0.5 (or at rest). By sliding it either left or right, we can make the hinge spin in either direction. The further you slide it, the faster it will spin in that direction.

NOTE: This is just an example of using the UI, and the hinge is more of a hybrid than a true motor. If you move the slider halfway in either direction, the hinge will try to act somewhat like a servo and somewhat like a motor. When the slider is fully left or right, it will act like a true motor. When precision is needed, use the appropriate joint for the job. (A motor or a servo.)

Another interesting thing we can do when setting the controller value is to **set multiple joints to the same controller name.** By doing this, you can control many joints at once with one slider. We'll see an example of this on the next page when we add steering to our original car model.

But before we get to that, let’s look at the final option in our UI: the damp setting. You can use this to smooth out "twitchy" joints and make them run more smoothly. Play around with different settings in the above model and see how it affects the model.

Remember, each joint type has its own settings in the UI, and the only way to know how any setting will affect that joint is to plug in some values and test it out.

Now that we have the basics of using the UI down, let's continue on with learning the UI and add some steering and forward/ reverse control to our original car model.

Continue on te Advanced User Interface

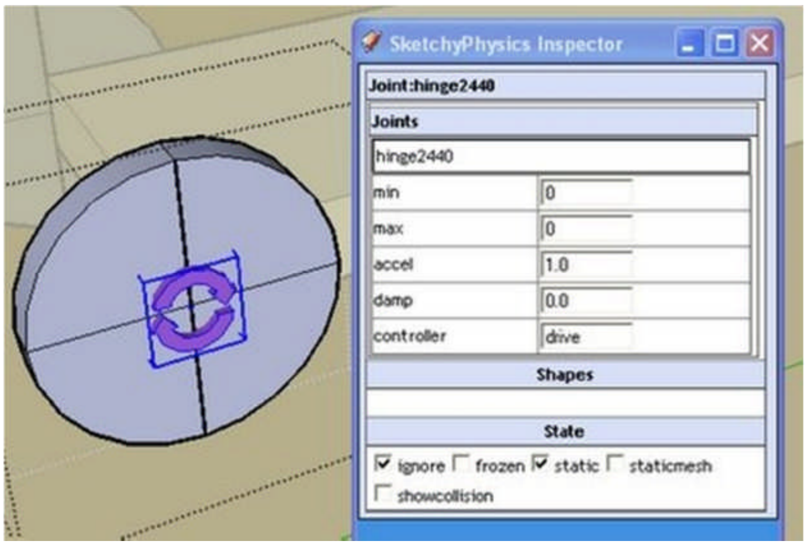
Advanced SketchyPhysics...using the UI Part 2

Now we have some knowledge of the UI, let's use it to finish up our original car model and add both steering and forward/ reverse capabilities.

Starting with our original model... if we wanted to make the car move, we could replace the original rear hinges with motors. This would spin the wheels, but only in one direction. In our case we want both forward AND reverse, so we'll set up our two rear hinges as custom motors using the UI.

(You can download this model HERE to follow along with the tutorial.)

We'll start by selecting one of the rear wheel hinges and setting the **accel** value in the UI to one. This will allow it to spin on it's own. Next we'll set the **controller** value to "drive".



SketchyPhysics Inspector

Joint:hinge2440

Joints

hinge2440

min	0
max	0
accel	1.0
damp	0.0
controller	drive

Shapes

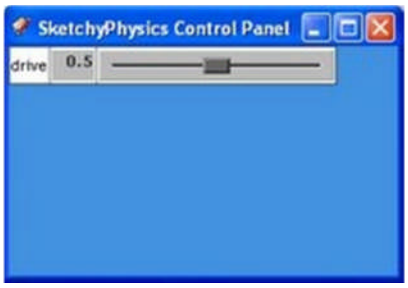
State

[✓ ignore] frozen [✓ static] staticmesh

showcollision

Repeat the same steps for the other rear wheel. ***Be sure to set the controller name exactly the same as the other***, so they will work together when controlled with one slider.

Test the model now. Use the "drive" slider and be sure the wheels are spinning in the same direction.



SketchyPhysics Control Panel

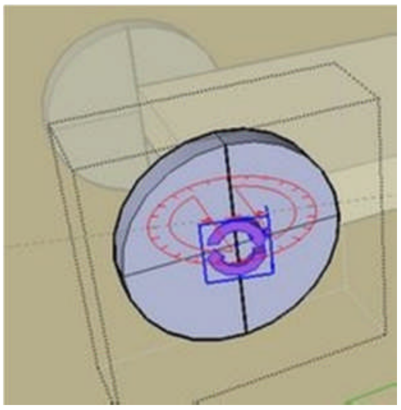
drive 0.5

If one wheel spins in the opposite direction as the other, edit your model and use the move tool to rotate one hinge completely around.

NOTE:

DO NOT use the "Flip" command shortcut! Using the flip command on parts of models to be used with SketchyPhysics usually will not work! Many models are completely messed up when you start "flipping" things. **Always rotate a hinge using the move tool!**

Rotating a hinge using the MOVE tool to make it spin the same direction as the wheel on the other side:

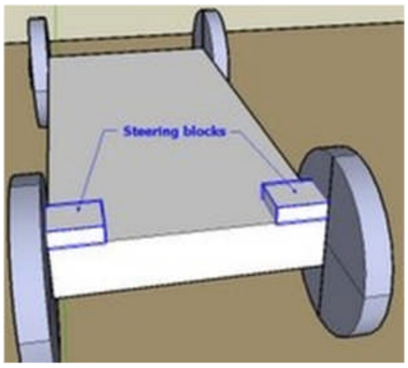


Test the model again and make sure your rear wheels are spinning in the same direction. Move the slider left and right. You should now have forward and reverse on your car! If the car is out of control, or the wheels don't spin evenly, play around with the damp settings or lower the accel setting. I have found an accel setting of 1.0 and a damp setting of 1.0 seems to work well in this model. Next comes the steering.

This is a little more complex than the drive because we'll need to add some more geometry to our model and also a few more joints.

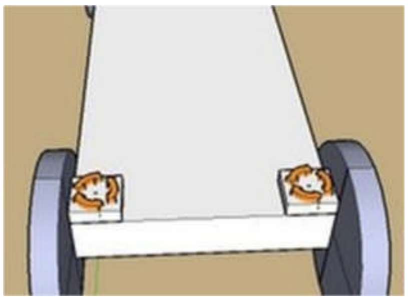
The basic concept is to place an object between the wheels and the body that will rotate horizontally.

The easiest way I have found is to add a pair of blocks to the front of the car.



NOTE: These blocks are SEPARATE groups. They are NOT part of the body or wheel groups, and they are NOT grouped to each-other!

Next we'll add a servo for each block. Group each servo and its block, then joint-connect each servo to the body.



Now we'll need to use the UI to set the steering servos to behave as needed. Set each servo's min and max to -45 and 45 to limit their swing. Also set their controller name the same (I'm using "steering") so they'll work together.

The next step is to make the wheels move with the steering blocks. So using the **joint connector tool**, break the original connection of each wheels hinge to the body (using SHIFT) and make a new connection to their steering block. (Using CTRL.)

The final thing we need to do is to modify the body so the wheels have some room to turn. (Wheel-wells.) Here's one way of doing this:

Sorry, no picture

Notice that the body is now made of two grouped objects grouped together. The main body and the nose of the body. If I simply push-pulled the original body and left it as one shape, the geometry would still reflect a rectangle and the wheels would hit the "invisible" body left over, unable to turn.

Be sure to use the "show collision" option in the UI to shape your body until it works right.

One downside of changing the original body to two groups grouped together, is that by re-grouping the original group, it broke any joint-connections that were associated with it originally. We'll look at some troubleshooting below.

This is usually the point where people get frustrated with the model because things bind up and parts fall off. So let's look at some possible reasons for failure.

When things go wrong...

Stay calm, debug everything, and check your joint connections.

When I changed the body from one group to two (grouped) groups, it broke the original joint connections. So I had to re-joint connect the rear hinges to the "new" body and also re-joint connect the servos to the body as well. Follow these next few steps when troubleshooting a stubborn model:

Anytime you explode a group, or group a group with another object (even a joint) the original joints connected to it will be broken!!!

If things don't move like they should, or it looks like joints are "stretching", they're probably binding against something. (Maybe something invisible...so DEBUG your geometry.)

If jointed parts fall off your model, remake the joint connections.

When all else fails, there is NO SHAME in deleting a joint completely and replacing it with a new one.

The above example was only one way to make the geometry work. I'm sure there are many more ways to make it better. A steering arm comes to mind instead of my "magic" floating steering blocks! But either way, we now have a car that can roll forward and reverse, and also steer.

If you want to reference the finished model, it can be found HERE.

Here's a tip when testing your car:

After you press RUN, and before you start moving it around, Right-click the body and set the camera to "FOLLOW". This will keep the car from running off your screen. So you can dedicate more time to troubleshooting, and less time zooming and panning to follow the car around.

Also note that this car is far from being complete, as all the joints need their accel and damp settings tweaked. So there is still a lot of work to be done, but now you should have the tools to make it through!