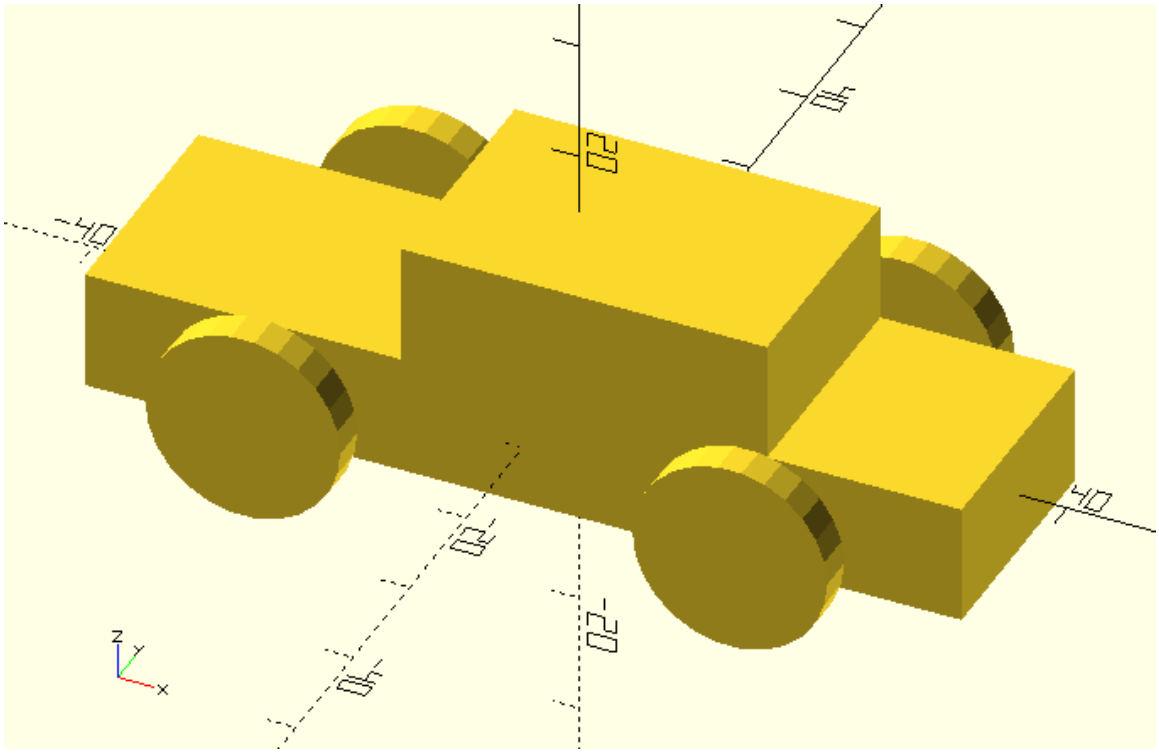
Chapter 2

Scaling parts or the whole model

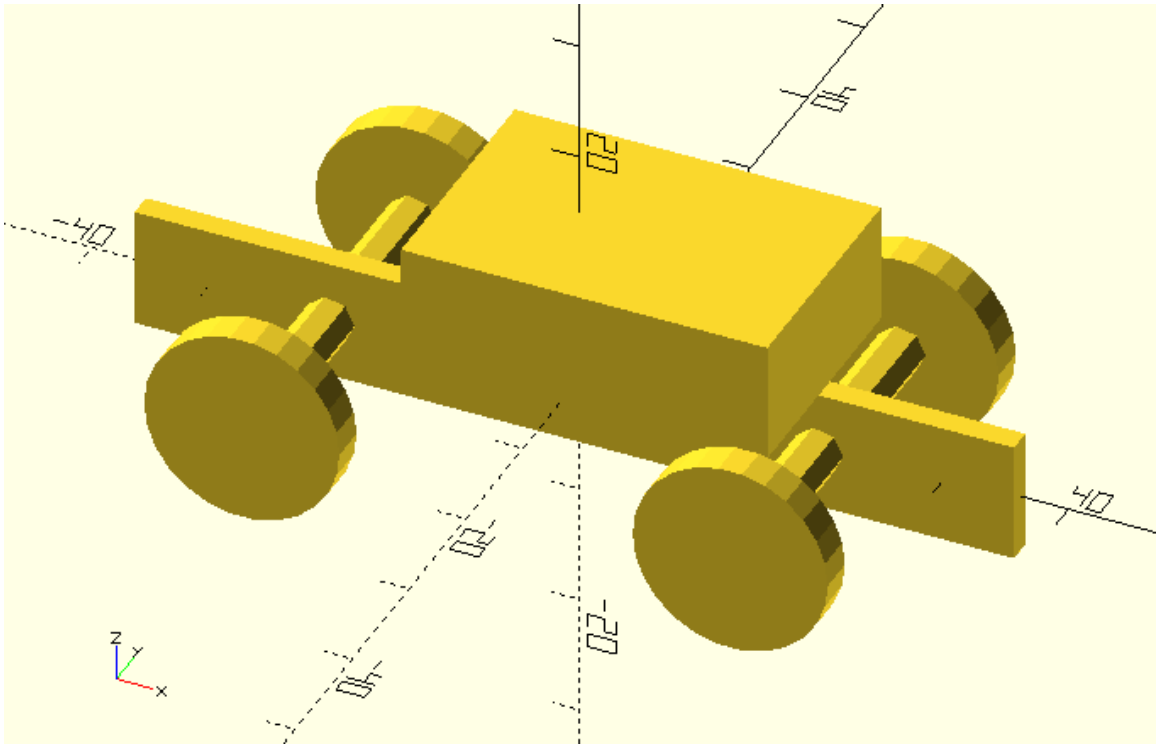
The car you just created is a pretty cool model considering it was you first time using OpenSCAD. However, you probably didn't put enough thought into the proportions of its different elements or its overall size. Don't worry though, you can use the scale command, which is another one of the transformation commands. Modify the statement that creates the base of the car's body in the following way in order to increase the length of the body by a ratio of 1.2.

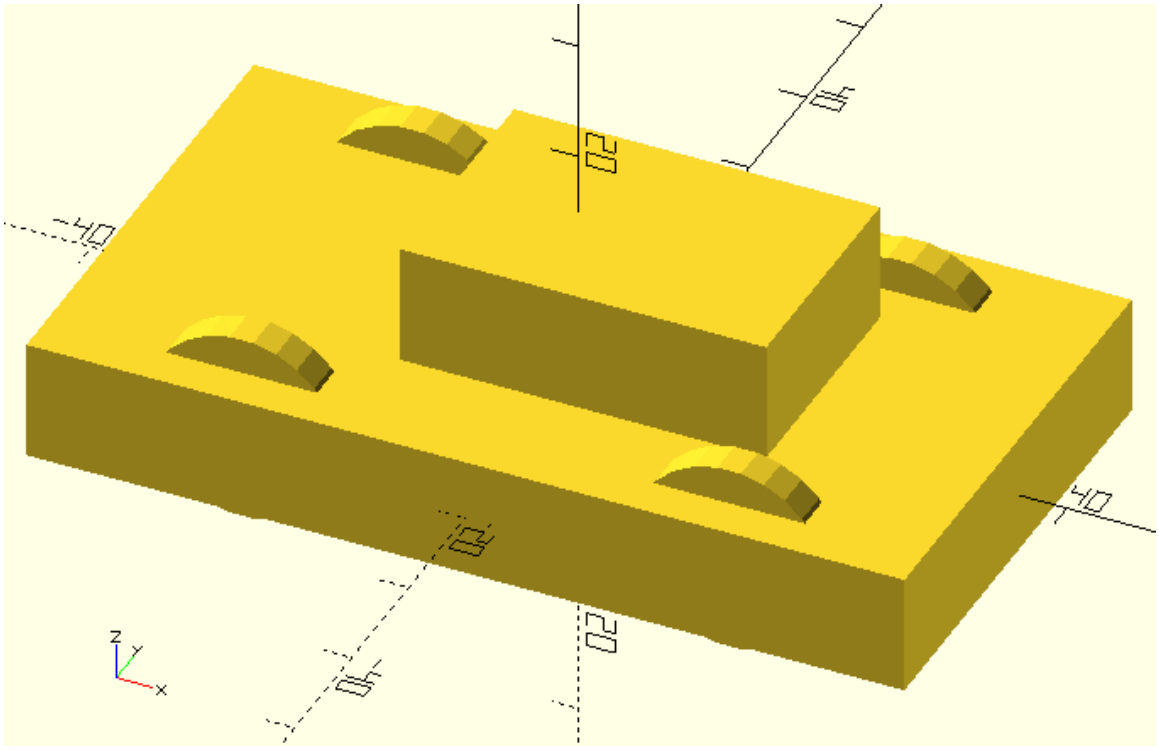
```
// Car body base

scale([1.2,1,1])cube([60,20,10],center=true);
```



You should notice that the scale command is used like the transform and rotate commands. It is added to the left of an existing statement without including a semicolon in between and it has a vector of three values as an input parameter. In analogy to the translate and rotate commands each value corresponds to the scaling ratio along the X, Y and Z axis.

Try modifying the input of the scale command in order to scale the base of the body by a factor of 1.2 along the X axis and a factor of 0.1 or 2 along the Y axis. Did you get anything that could be a Mars rover or a tank? Are you surprised with how different the models look compared to the original car?

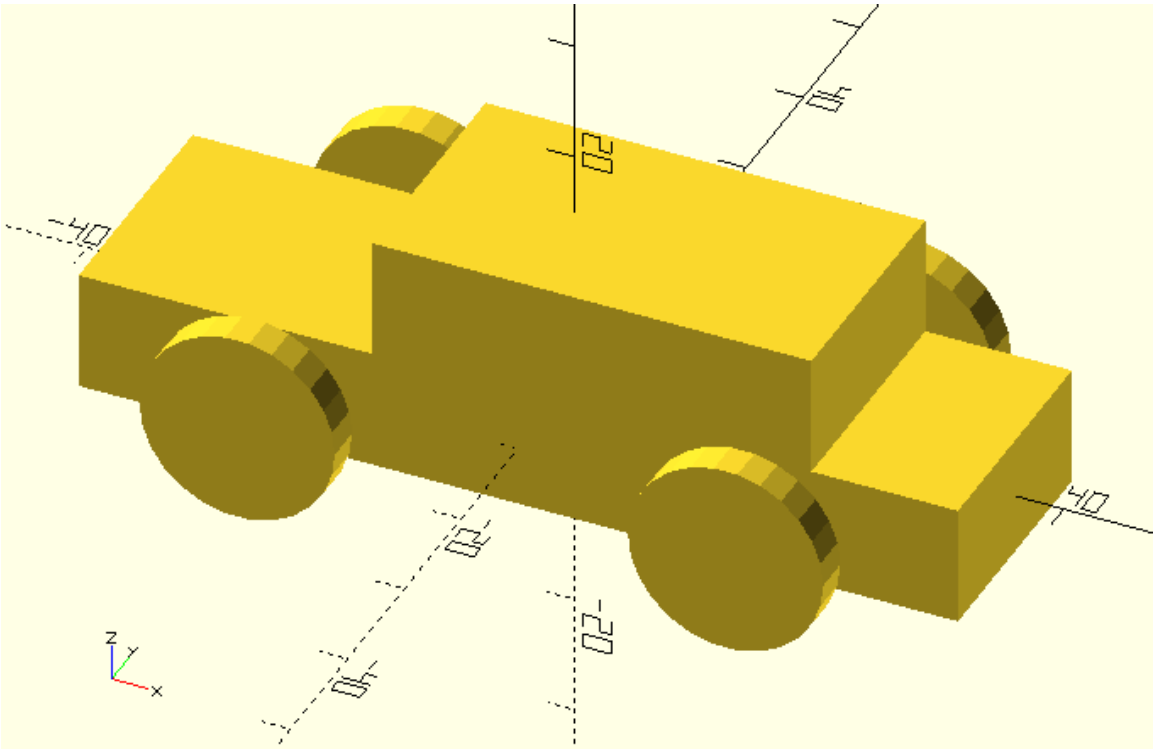




It is also possible to apply the same scale command or any other transformation command to more than one objects. Use the following code to apply the scale command to both the base and the top of the car's body.

```
scale([1.2,1,1]){
```

```
// Car body base

cube([60,20,10],center=true);

// Car body top

translate([5,0,10])cube([30,20,10],center=true);

}
```



The first thing you should notice is that in order to apply the scale command to more than one object, a set of curly brackets it used. The statements that define the corresponding objects along with their semicolons are placed inside the curly brackets. The curly brackets don't require a semicolon at the end.

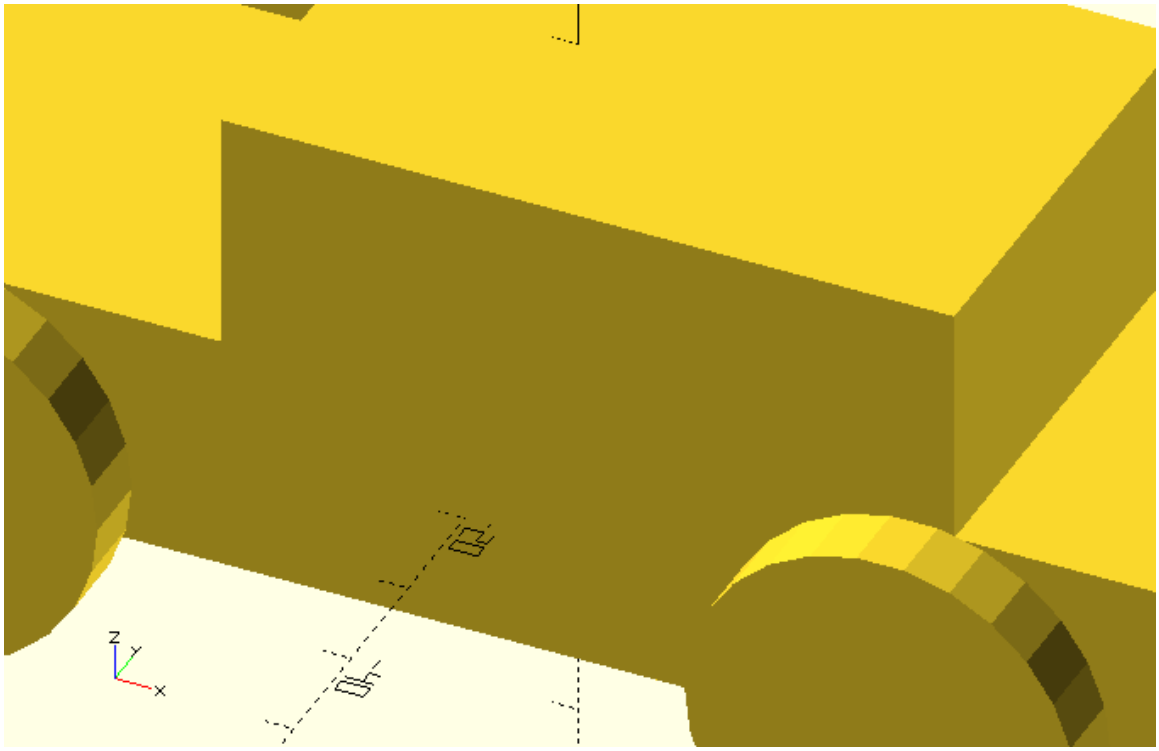
The second thing you should notice is how the use of white space and comments increase the readability of your script. The following script is exactly equivalent, you can decide for yourself which one you'd rather have to read.

```
scale([1.2,1,1]){cube([60,20,10],center=true);

translate([5,0,10])cube([30,20,10],center=true);}
```

Try applying the scale command to your whole model. Did you remember to include all statements inside the curly brackets? What should be the relation between the scaling factors along the X and Z axis so that the wheels don't deform? What should the scaling factors be to get a car that has the same proportions but double the size?

*For the wheels not to deform, the scaling factors along the X and Z axis should be equal.

```
scale([2,2,2]){
// Car body base
cube([60,20,10],center=true);
// Car body top
translate([5,0,10])cube([30,20,10],center=true);
// Front left wheel
translate([-20,-15,0])rotate([90,0,0])cylinder(h=3,r=8,center=true);
// Front right wheel
translate([-20,15,0])rotate([90,0,0])cylinder(h=3,r=8,center=true);
// Rear left wheel
translate([20,-15,0])rotate([90,0,0])cylinder(h=3,r=8,center=true);
// Rear right wheel
translate([20,15,0])rotate([90,0,0])cylinder(h=3,r=8,center=true);
// Front axle
translate([-20,0,0])rotate([90,0,0])cylinder(h=30,r=2,center=true);
// Rear axle
translate([20,0,0])rotate([90,0,0])cylinder(h=30,r=2,center=true);
}
```



Quick quiz

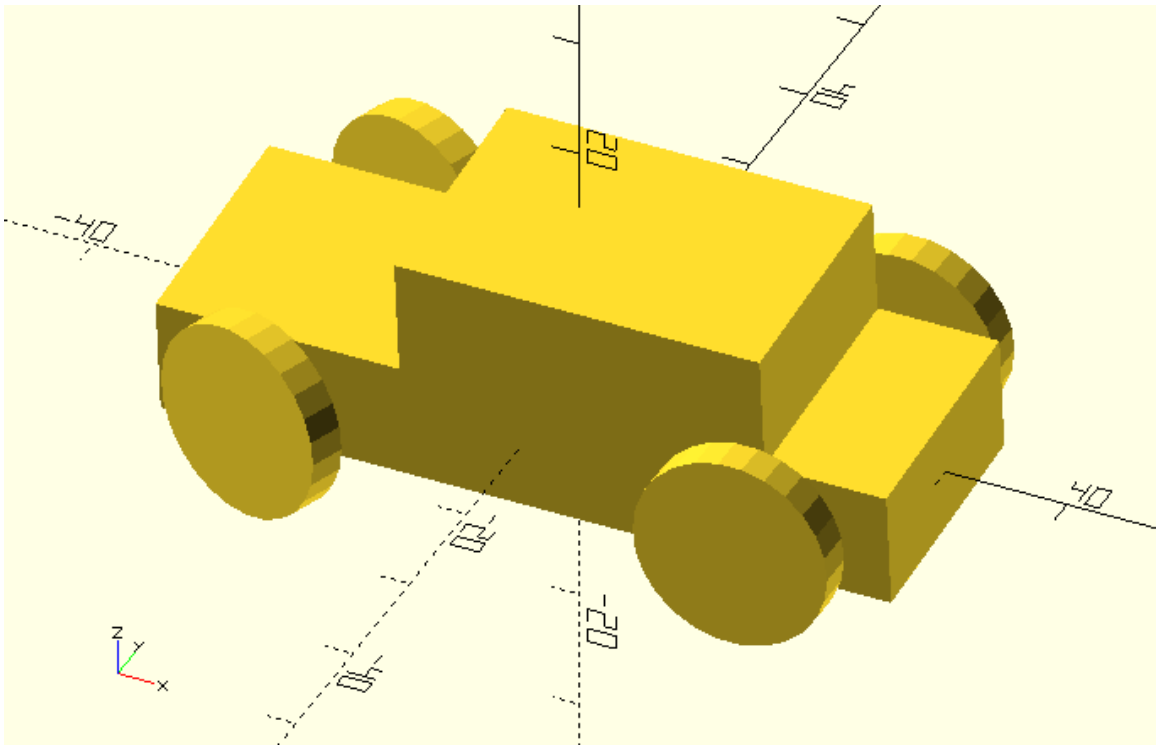
The following script is the model you created in the first chapter.

```
// Car body base
cube([60,20,10],center=true);
// Car body top
translate([5,0,10])cube([30,20,10],center=true);
// Front left wheel
translate([-20,-15,0])rotate([90,0,0])cylinder(h=3,r=8,center=true);
// Front right wheel
translate([-20,15,0])rotate([90,0,0])cylinder(h=3,r=8,center=true);
// Rear left wheel
translate([20,-15,0])rotate([90,0,0])cylinder(h=3,r=8,center=true);
// Rear right wheel
translate([20,15,0])rotate([90,0,0])cylinder(h=3,r=8,center=true);
// Front axle
translate([-20,0,0])rotate([90,0,0])cylinder(h=30,r=2,center=true);
```

```
// Rear axle
translate([20,0,0])rotate([90,0,0])cylinder(h=30,r=2,center=true);
```

Try rotating the front wheels by 20 degrees around the Z axis, as if the car was making a right turn. In order to make your model more convincing, try rotating the body of the car (base and top) by 5 degrees around the X axis in the opposite direction of the turn. To turn the wheels, modify the input parameters of existing rotate commands, to turn the body add a new rotate command.

```
rotate([5,0,0]){
// Car body base
cube([60,20,10],center=true);
// Car body top
translate([5,0,10])cube([30,20,10],center=true);
}
// Front left wheel
translate([-20,-15,0])rotate([90,0,-20])cylinder(h=3,r=8,center=true);
// Front right wheel
translate([-20,15,0])rotate([90,0,-20])cylinder(h=3,r=8,center=true);
// Rear left wheel
translate([20,-15,0])rotate([90,0,0])cylinder(h=3,r=8,center=true);
// Rear right wheel
translate([20,15,0])rotate([90,0,0])cylinder(h=3,r=8,center=true);
// Front axle
translate([-20,0,0])rotate([90,0,0])cylinder(h=30,r=2,center=true);
// Rear axle
translate([20,0,0])rotate([90,0,0])cylinder(h=30,r=2,center=true);
```

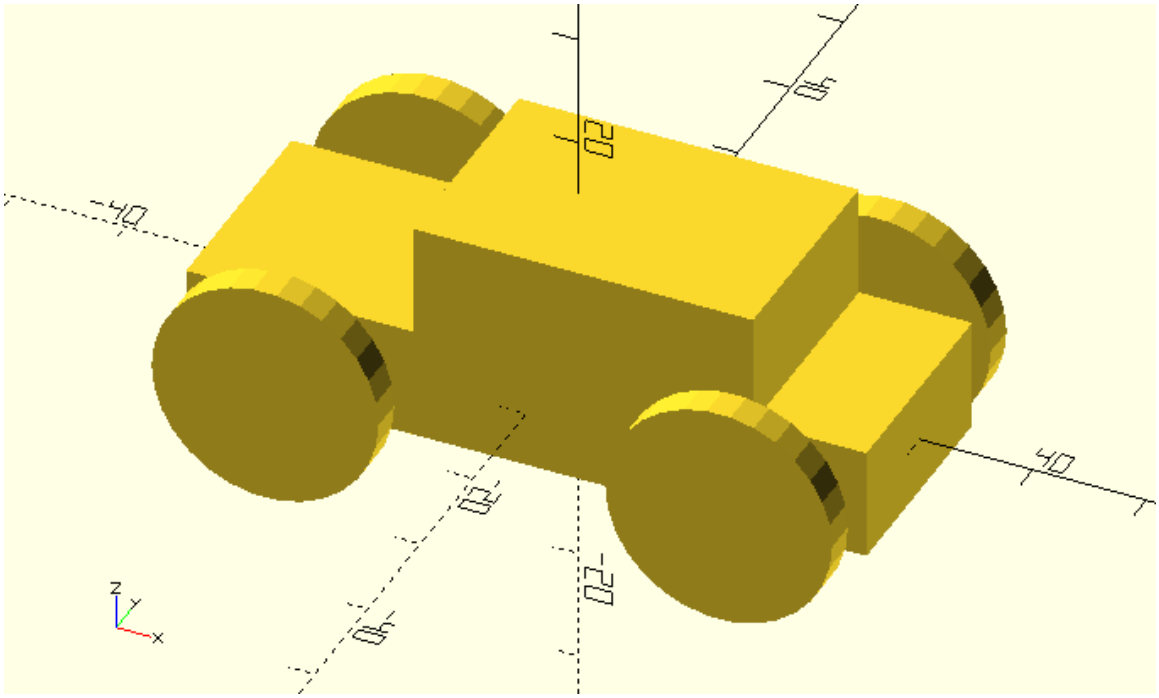


Parameterizing parts of your model

You should have gotten the point that a model is most of the times not intended to exist in one version. One of the powers of OpenSCAD scripting language lies in making easy the ability to reuse models over and over again or simply to play around with them until you are satisfied to commit to a final version. It's time to make some modifications to your car!

Try changing the diameter of the wheels to 10 units. How easily did you find which values to modify? Did you have to do the same thing four times?

```
// Front left wheel

translate([-20,-15,0])rotate([90,0,0])cylinder(h=3,r=10,center=true);

// Front right wheel

translate([-20,15,0])rotate([90,0,0])cylinder(h=3,r=10,center=true);

// Rear left wheel

translate([20,-15,0])rotate([90,0,0])cylinder(h=3,r=10,center=true);

// Rear right wheel

translate([20,15,0])rotate([90,0,0])cylinder(h=3,r=10,center=true);
```



Although it wasn't that hard to change the size of wheels it could have been much simpler. First, it could have been easier to find which values to change. Second, you could have only one value to change since all wheels have the same radius. All this can be achieved with the use of variables. In the following script a variable for the radius of the wheels is introduced.

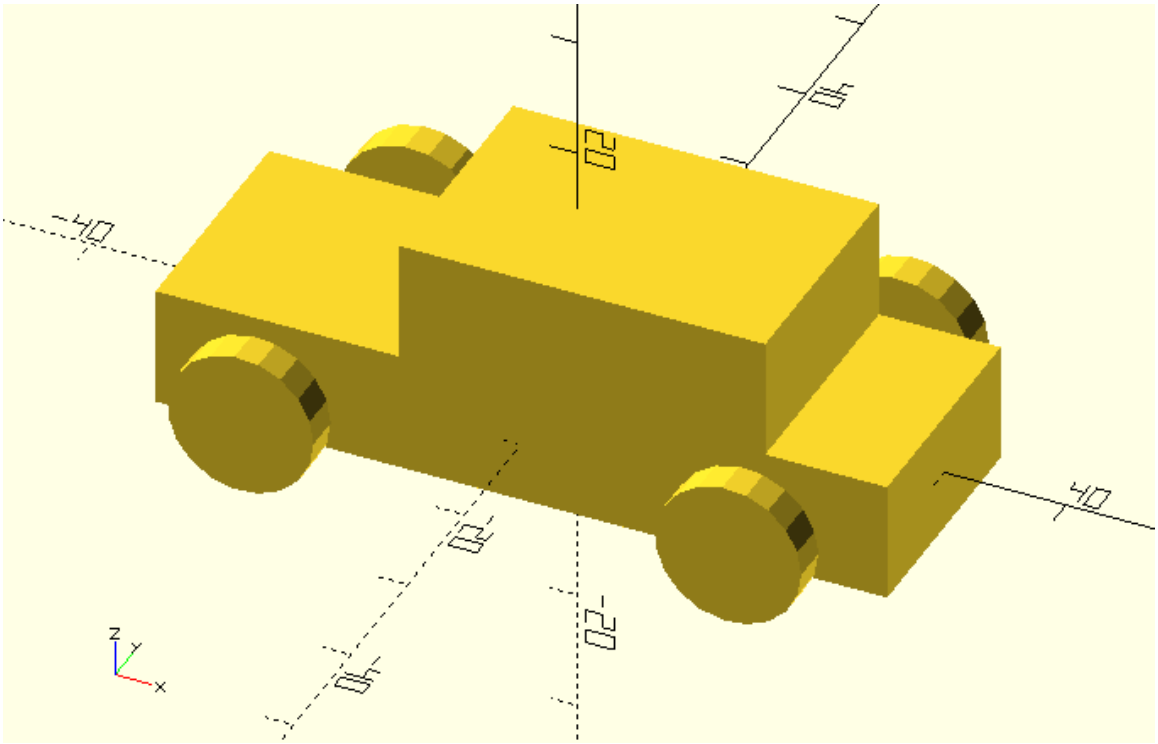
```
wheel_radious = 8;

// Front left wheel

translate([-20,-15,0])rotate([90,0,0])cylinder(h=3,r=wheel_radious,center=true);

// Front right wheel

translate([-20,15,0])rotate([90,0,0])cylinder(h=3,r=wheel_radious,center=true);

// Rear left wheel

translate([20,-15,0])rotate([90,0,0])cylinder(h=3,r=wheel_radious,center=true);

// Rear right wheel

translate([20,15,0])rotate([90,0,0])cylinder(h=3,r=wheel_radious,center=true);
```

The first thing you should notice is how a variable should be defined. First, the name of the variable needs to be typed out. In this example the variable is named wheel_radius. A valid variable name should ..... Then the equal sign followed by the value that you would like to assign to that variable needs to be typed out. Finally, a semicolon is required at the end to denote the completion of that statement. It's a good practice to keep your variables organized by defining them all at the top of the document.

The second thing you should notice is the modified input to the cylinder commands. The input parameter r has been set equal to the corresponding variable that has previously been defined. When OpenSCAD evaluates the script, it sets the input parameter r equal to the value of the wheel_radius variable.

Try using a variable named wheel_radius to define the size of your car's wheels. Try changing the size of the wheels a few times by modifying the value of the wheel_radius variable. How easier did you find changing the size of the wheels using the wheel_radius variable?

```
wheel_radious = 6;

// Car body base

cube([60,20,10],center=true);

// Car body top

translate([5,0,10])cube([30,20,10],center=true);

// Front left wheel

translate([-20,-15,0])rotate([90,0,0])cylinder(h=3,r=wheel_radious,center=true);

// Front right wheel

translate([-20,15,0])rotate([90,0,0])cylinder(h=3,r=wheel_radious,center=true);

// Rear left wheel

translate([20,-15,0])rotate([90,0,0])cylinder(h=3,r=wheel_radious,center=true);

// Rear right wheel

translate([20,15,0])rotate([90,0,0])cylinder(h=3,r=wheel_radious,center=true);

// Front axle

translate([-20,0,0])rotate([90,0,0])cylinder(h=30,r=2,center=true);

// Rear axle

translate([20,0,0])rotate([90,0,0])cylinder(h=30,r=2,center=true);
```

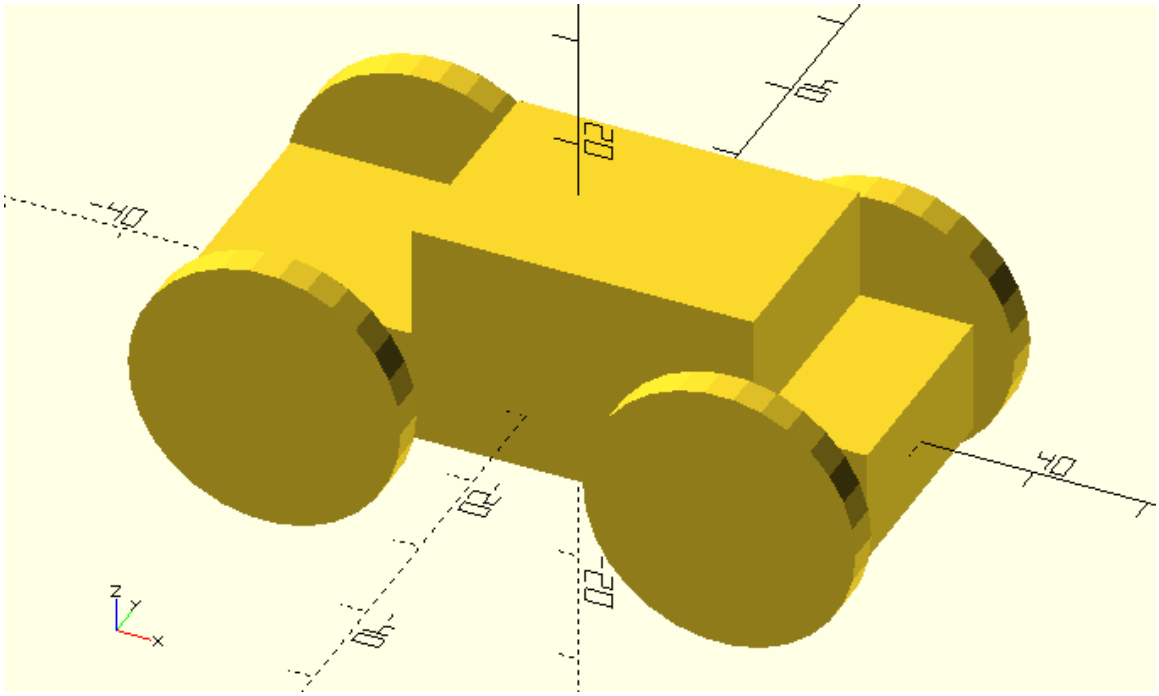


There is one important thing you should keep in mind about the behavior of variables in OpenSCAD. The variables in OpenSCAD behave like constants. They can hold only one value which they keep throughout the creation of your model. So, what happens if you assign a value to wheel_radius at the start of your script and then assign a new value to it after the definition of the two front wheels? Will the rear wheels have different size compared to the front wheels?

Try assigning a different value to the wheel_radius variable right after the definition of the front wheels. Does your car have different front and rear wheel size?

```
wheel_radius = 6;

// Car body base

cube([60,20,10],center=true);

// Car body top

translate([5,0,10])cube([30,20,10],center=true);

// Front left wheel

translate([-20,-15,0])rotate([90,0,0])cylinder(h=3,r=wheel_radius,center=true);

// Front right wheel

translate([-20,15,0])rotate([90,0,0])cylinder(h=3,r=wheel_radius,center=true);

wheel_radius = 12;
```

```
// Rear left wheel

translate([20,-15,0])rotate([90,0,0])cylinder(h=3,r=wheel_radius,center=true);

// Rear right wheel

translate([20,15,0])rotate([90,0,0])cylinder(h=3,r=wheel_radius,center=true);

// Front axle

translate([-20,0,0])rotate([90,0,0])cylinder(h=30,r=2,center=true);

// Rear axle

translate([20,0,0])rotate([90,0,0])cylinder(h=30,r=2,center=true);
```



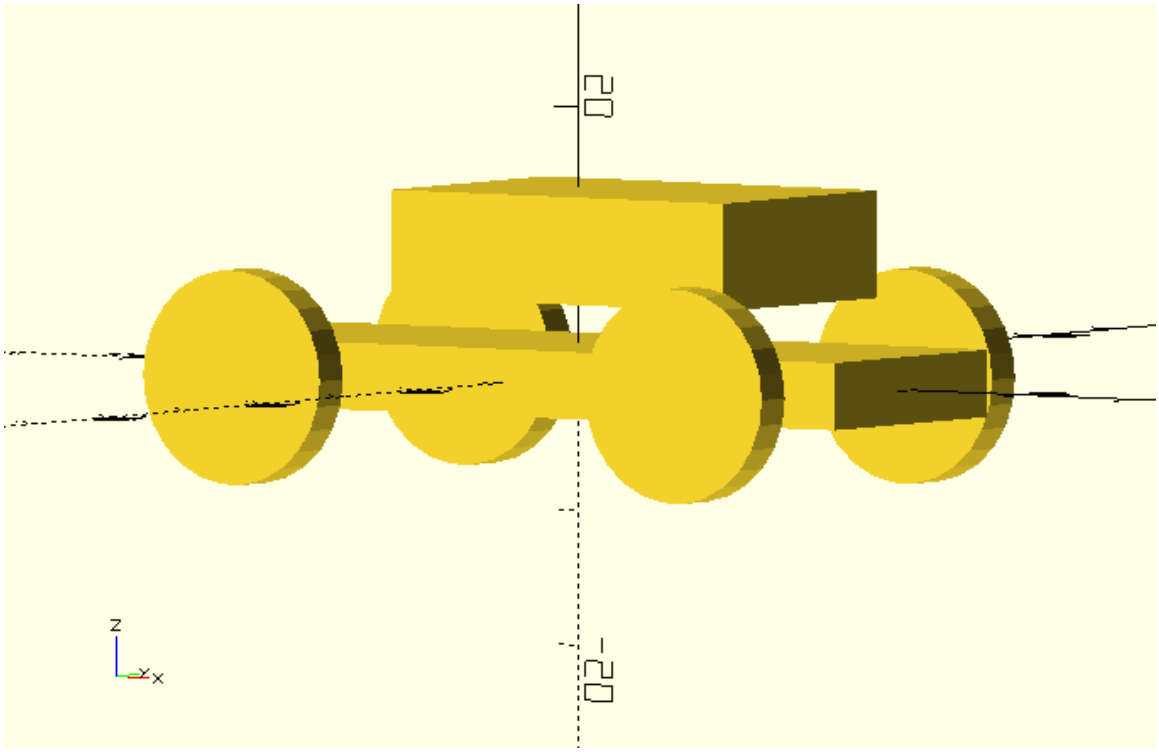
You should notice that all wheels have the same size. If multiple assignments to a variable exist, OpenSCAD uses the value of the last assignment. Even statements that make use of this variable and are defined before the last value assignment, will use the value of the last assignment.

Parameterizing more parts of your model

You can now easily play around with the size of the wheels. It would be nice if you were able to customize more aspects of your model with such ease. You should notice for a moment that modifying the size of the wheels doesn't affect any other aspect of your model, it doesn't break your model in any way. This is not always the case.

Try modifying the height of the car's body base and top by defining a base_height and a top_height variable and making the appropriate changes to the corresponding statements that define the base and the top. Assign the value 5 to the base_height variable and the value 8 to the top_height variable. What do you notice?

```
base_height = 5;

top_height = 8;

// Car body base

cube([60,20,base_height],center=true);

// Car body top

translate([5,0,10])cube([30,20,top_height],center=true);
```

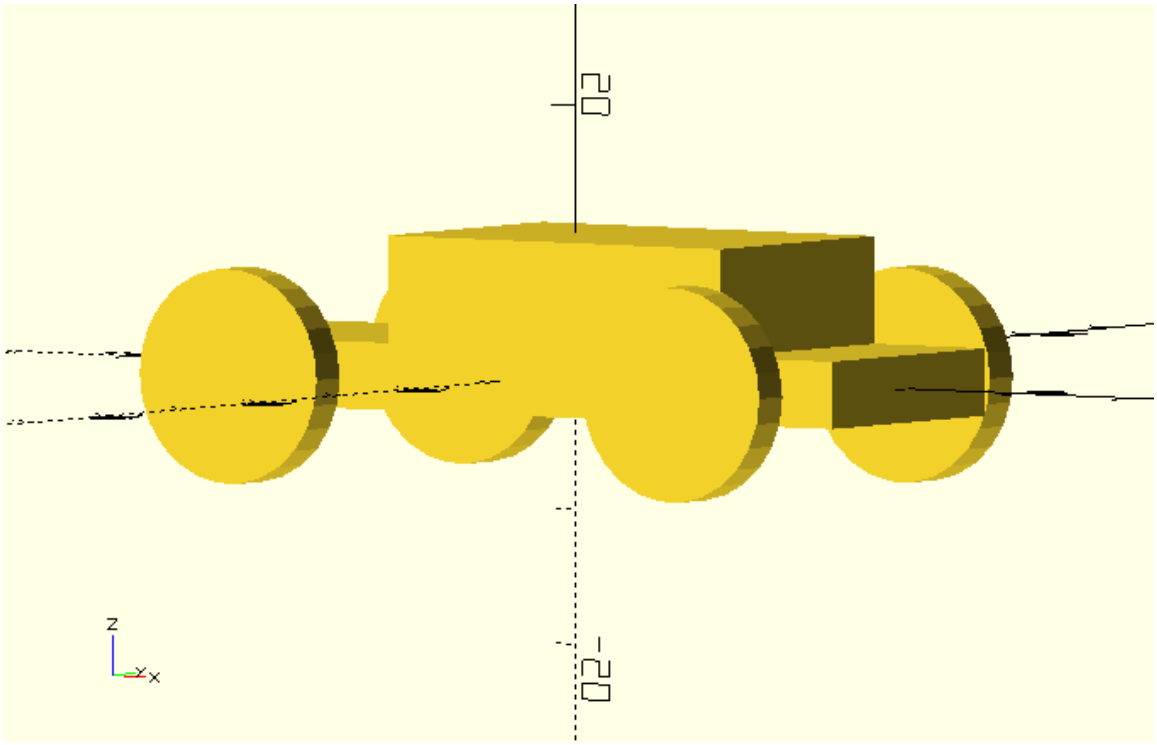


It is obvious that the body of the car stops being one as the base and the top separate. This happened because the correct position of the body's top is dependent on the height of the body's base and the height of body's top. Remember that in order to make the top sit on top of the base you had to translate the top along the Z axis by an amount equal to half the height of the base plus half the height of the top. If you want to parameterize the height of the base and the top you should also parameterize the translation of the top along the Z axis.

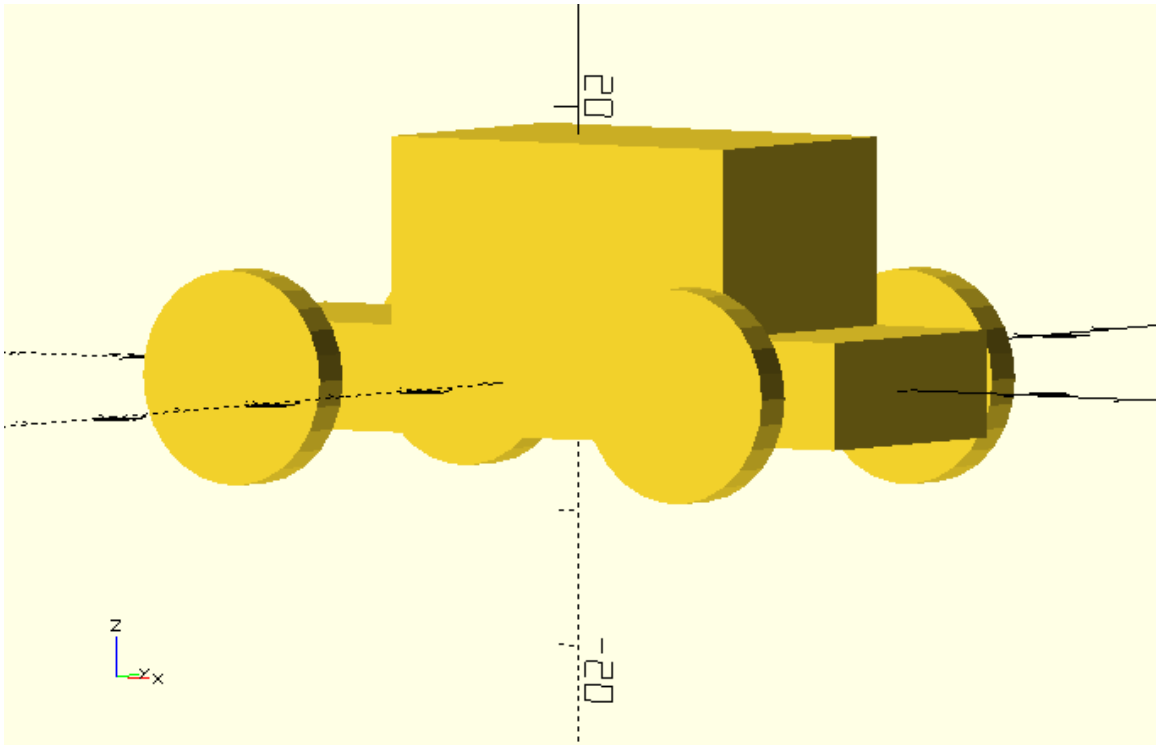
Try parameterizing the translation of the body's top along the Z axis using the base_height and base_top variables to make it sit on top of the body's base. Try assigning different values to the base_height and top_height variables. Does the position of the body's top remain correct?

```
base_height = 5;

top_height = 8;
```

```
wheel_radius = 8;

// Car body base

cube([60,20,base_height],center=true);

// Car body top

translate([5,0,base_height/2+top_height/2])cube([30,20,top_height],center=true);
```



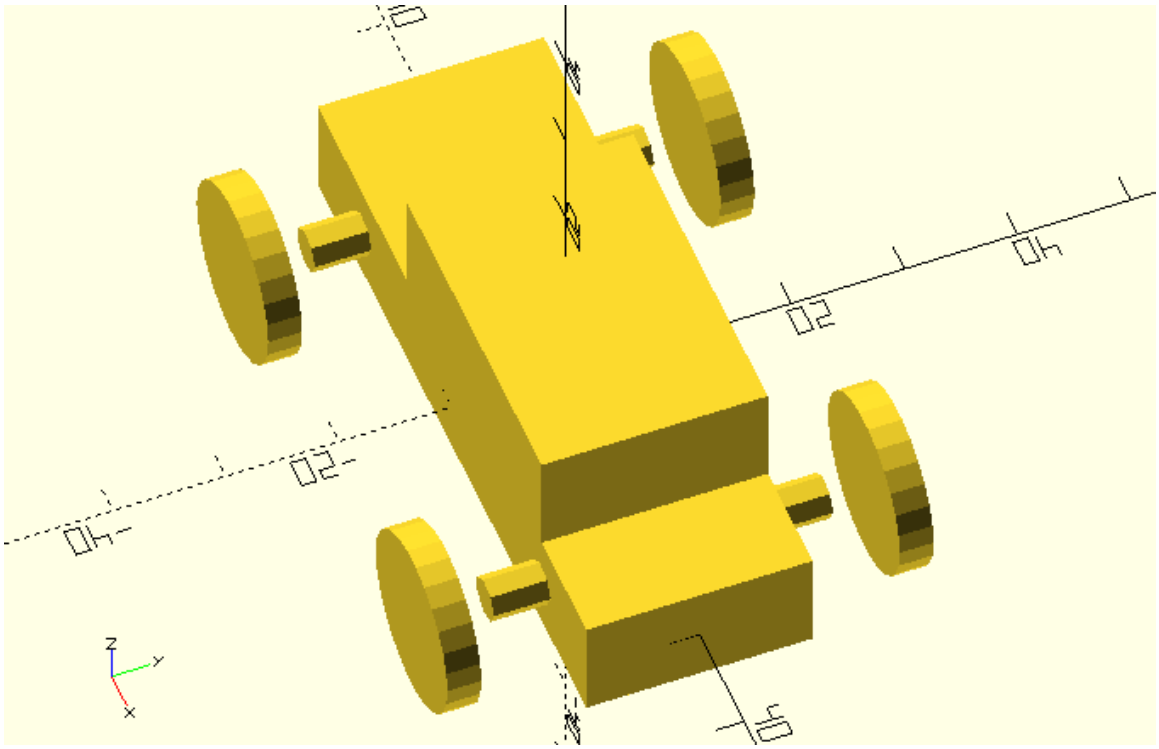

```
base_height = 8;

top_height = 14;
```



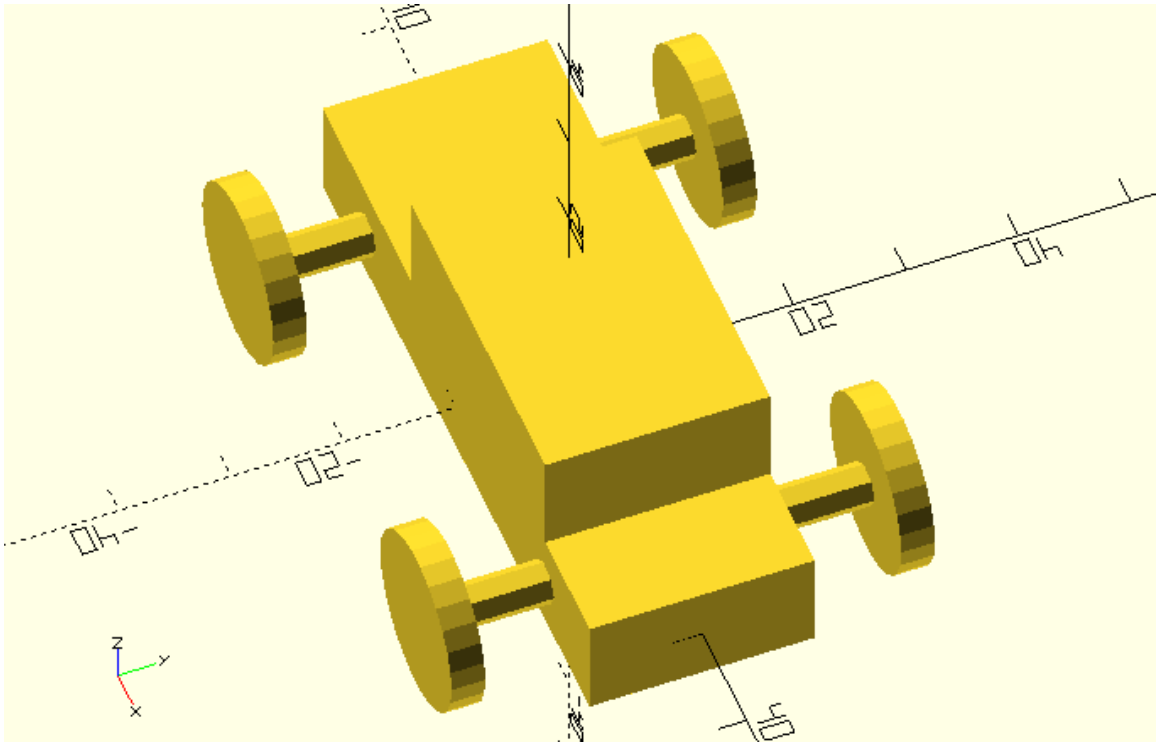
You should remember that every time you parameterize some aspect of your model you should also parameterize additional dependent aspects to prevent your model from breaking apart.

Try parameterizing the track (lateral distance of the wheels) using a new variable named track. Try assigning different values to the track variable. What do you notice? Does any other aspect of your model depend on the value of the track variable? If yes, use the track variable to parameterize it so your model doesn't break apart.

```
track = 40;

// Front left wheel

translate([-20,-track/2,0])rotate([90,0,0])cylinder(h=3,r=wheel_radius,center=true);

// Front right wheel

translate([-20,track/2,0])rotate([90,0,0])cylinder(h=3,r=wheel_radius,center=true);

// Rear left wheel

translate([20,-track/2,0])rotate([90,0,0])cylinder(h=3,r=wheel_radius,center=true);

// Rear right wheel

translate([20,track/2,0])rotate([90,0,0])cylinder(h=3,r=wheel_radius,center=true);
```



```
track = 40;
// Front left wheel
translate([-20,-track/2,0])rotate([90,0,0])cylinder(h=3,r=wheel_radius,center=true);
// Front right wheel
translate([-20,track/2,0])rotate([90,0,0])cylinder(h=3,r=wheel_radius,center=true);
// Rear left wheel
translate([20,-track/2,0])rotate([90,0,0])cylinder(h=3,r=wheel_radius,center=true);
// Rear right wheel
translate([20,track/2,0])rotate([90,0,0])cylinder(h=3,r=wheel_radius,center=true);
// Front axle
translate([-20,0,0])rotate([90,0,0])cylinder(h=track,r=2,center=true);
// Rear axle
translate([20,0,0])rotate([90,0,0])cylinder(h=track,r=2,center=true);
```

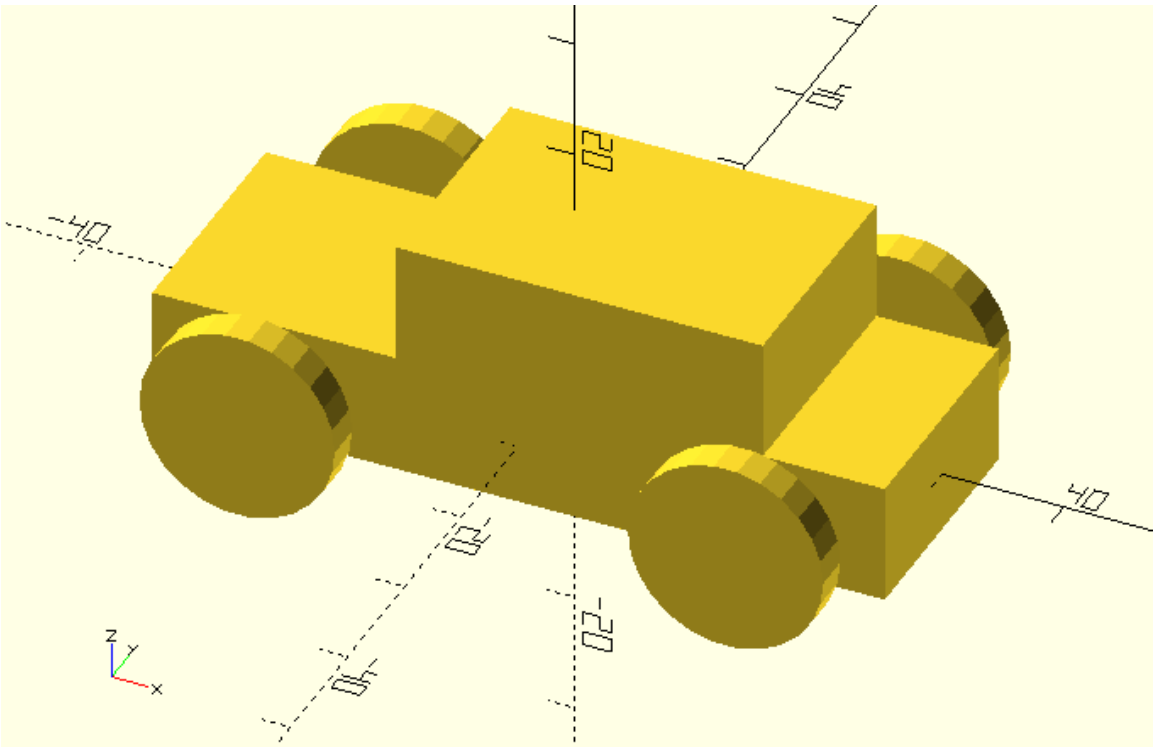


Challenge

The following script corresponds to the car model with parameterized wheel radius, base height, top height and track.

```
wheel_radius = 8;

base_height = 10;

top_height = 10;

track = 30;

// Car body base

cube([60,20,base_height],center=true);

// Car body top

translate([5,0,base_height/2+top_height/2])cube([30,20,top_height],center=true);

// Front left wheel

translate([-20,-track/2,0])rotate([90,0,0])cylinder(h=3,r=wheel_radius,center=true);

// Front right wheel

translate([-20,track/2,0])rotate([90,0,0])cylinder(h=3,r=wheel_radius,center=true);

// Rear left wheel
```

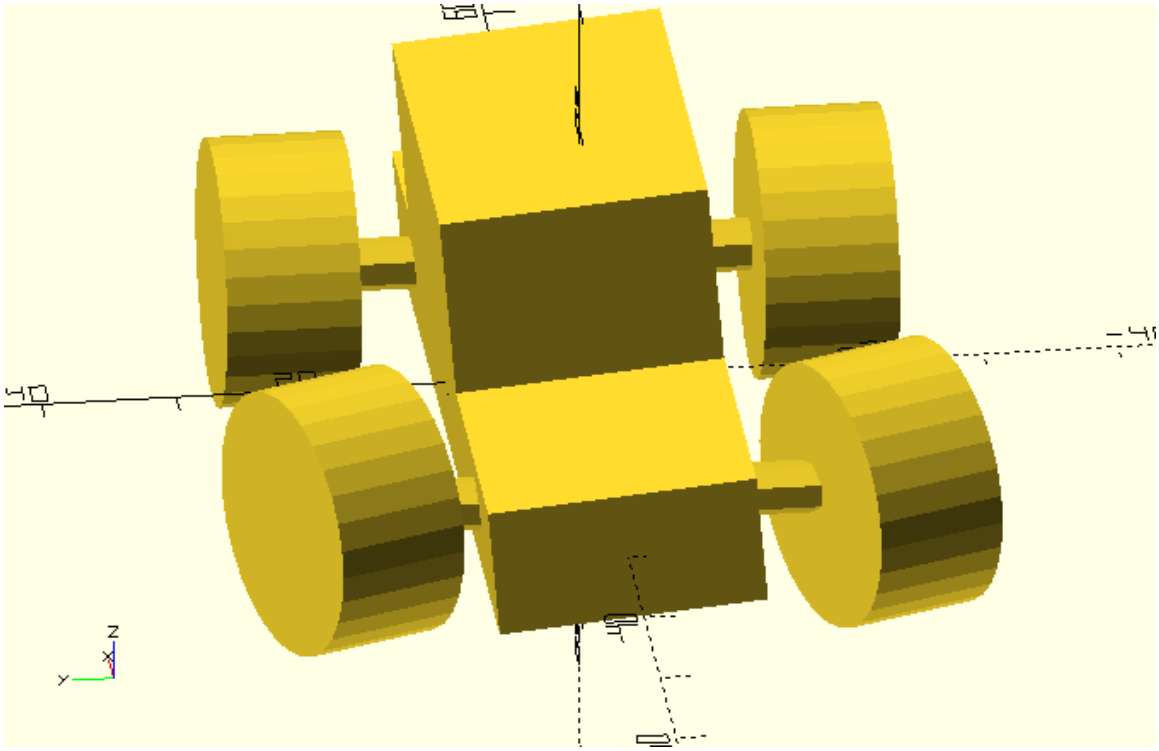
```
translate([20,-track/2,0])rotate([90,0,0])cylinder(h=3,r=wheel_radius,center=true);

// Rear right wheel

translate([20,track/2,0])rotate([90,0,0])cylinder(h=3,r=wheel_radius,center=true);

// Front axle

translate([-20,0,0])rotate([90,0,0])cylinder(h=track,r=2,center=true);

// Rear axle

translate([20,0,0])rotate([90,0,0])cylinder(h=track,r=2,center=true);
```



Try using a wheel_width variable to parameterize the width of the wheels, a wheels_turn variable to parameterize the rotation of the front wheels around the Z axis and a body_roll variable to parameterize the rotation of the body around the X axis. Experiment with assigning different values to wheel_radius, base_height, top_height, track, wheel_width, wheels_turn and body_roll to create a version of the car that you like.

```
wheel_radius = 10;

base_height = 10;

top_height = 14;

track = 40;

wheel_width = 10;
```

```
body_roll = -5;

wheels_turn = 20;

rotate([body_roll,0,0]){

// Car body base

cube([60,20,base_height],center=true);

// Car body top

translate([5,0,base_height/2+top_height/2])cube([30,20,top_height],center=true);

}

// Front left wheel

translate([-20,-
track/2,0])rotate([90,0,wheels_turn])cylinder(h=wheel_width,r=wheel_radius,center=true);

// Front right wheel

translate([-
20,track/2,0])rotate([90,0,wheels_turn])cylinder(h=wheel_width,r=wheel_radius,center=true);

// Rear left wheel

translate([20,-track/2,0])rotate([90,0,0])cylinder(h=wheel_width,r=wheel_radius,center=true);

// Rear right wheel

translate([20,track/2,0])rotate([90,0,0])cylinder(h=wheel_width,r=wheel_radius,center=true);

// Front axle

translate([-20,0,0])rotate([90,0,0])cylinder(h=track,r=2,center=true);

// Rear axle

translate([20,0,0])rotate([90,0,0])cylinder(h=track,r=2,center=true);
```



By now it should be clear to you that parameterizing your models unlocks the power of reusing, customizing and iterating your designs as well as that of effortlessly exploring different possibilities.

Parameterizing your own models

Have you put your new skills into use? Have you created any other models yourself?

Try parameterizing a few aspects or more of the models that you have created. See how far you can go! Experiment with assigning various combinations of values to the variables that you have defined. See how different the versions of your designs can be.