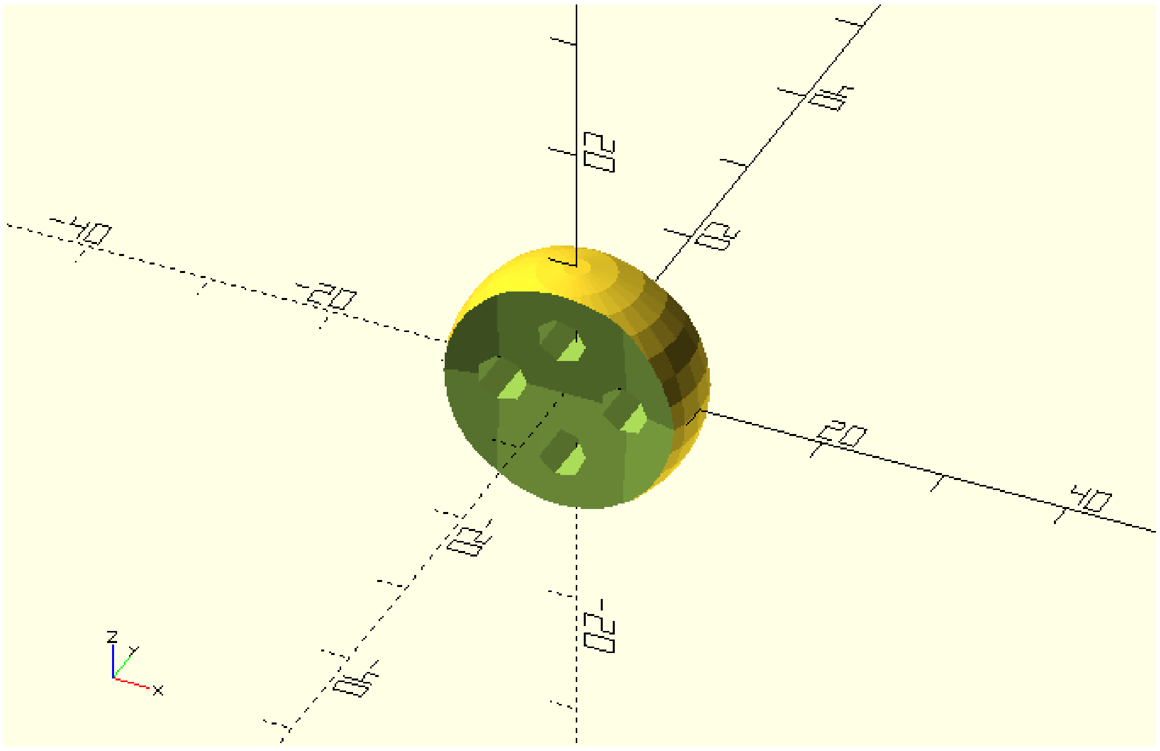
Chapter 4

Defining and using modules

The script of the last example of the previous chapter got quite long. This was the result of replacing the simple cylindrical wheels (which require one statement to be created) with a more complex wheel design (which requires many statements to be created). To change the wheels from the simple to the complex design you have to identify all cylinder commands that define the simple wheels and replace them with commands that define the complex wheels. This process sounds similar to the steps you had to go through to change the diameter of the wheels. When no use of variables was made you had to identify the corresponding values in your script and replace them one by one with the new value. This repetitive and time-consuming process was improved with the use of a wheel_radius variable which enabled you to quickly and easily change the diameter of the wheel. Can you do anything though to improve the corresponding error-prone process when you want to change completely the design of the wheels? The answer is yes! You can use modules which is the analogous of variables applied to whole parts/models. You can define a part of your design or even your whole model as a module.

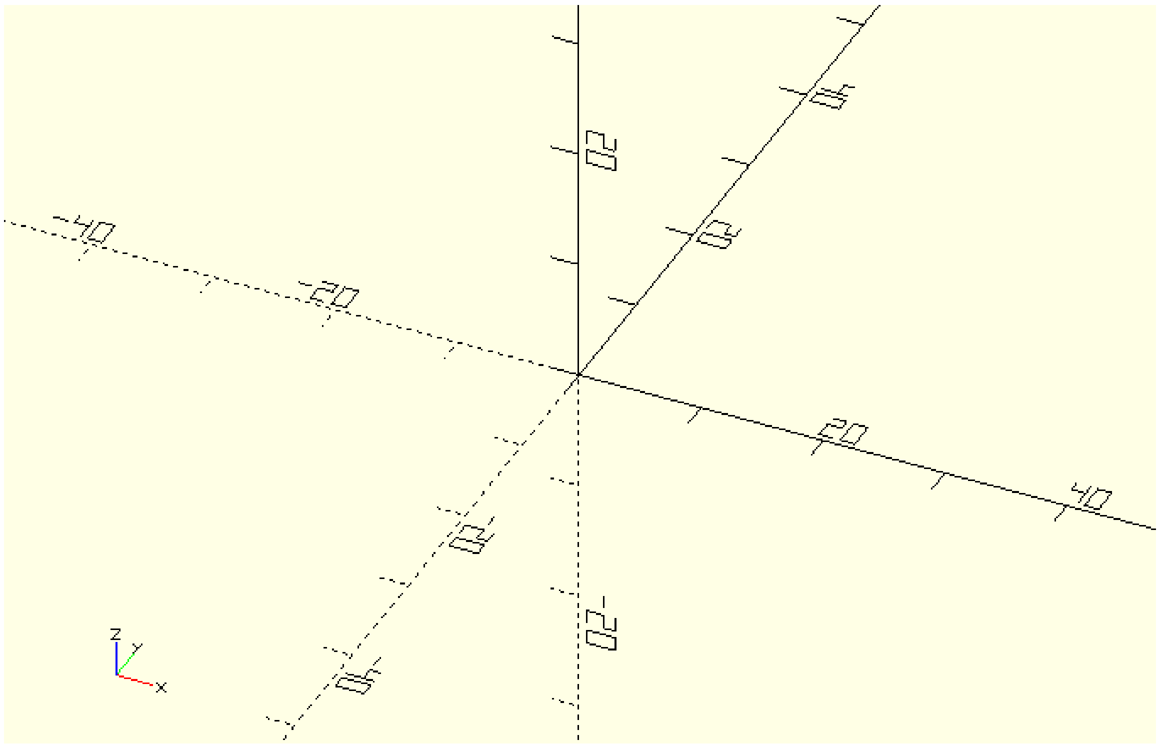
First remember for a moment the design of the complex wheel.

```
wheel_radius=10;

side_spheres_radius=50;

hub_thickness=4;

cylinder_radius=2;

cylinder_height=2*wheel_radius;

difference(){

// Wheel sphere

sphere(r=wheel_radius);

// Side sphere 1

translate([0,side_spheres_radius + hub_thickness/2,0])sphere(r=side_spheres_radius);

// Side sphere 2

translate([0,- (side_spheres_radius + hub_thickness/2),0])sphere(r=side_spheres_radius);

// Cylinder 1

translate([wheel_radius/2,0,0])rotate([90,0,0])cylinder(h=cylinder_height,r=cylinder_radius,center=true);

// Cylinder 2

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

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

// Cylinder 4

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

}
```



You can define the above wheel as a module in the following way.

```
 module wheel() {

wheel_radius=10;

side_spheres_radius=50;

hub_thickness=4;

cylinder_radius=2;

cylinder_height=2*wheel_radius;

difference(){

// Wheel sphere

sphere(r=wheel_radius);
// Side sphere 1

translate([0,side_spheres_radius + hub_thickness/2,0])sphere(r=side_spheres_radius);

// Side sphere 2

translate([0,- (side_spheres_radius + hub_thickness/2),0])sphere(r=side_spheres_radius);

// Cylinder 1

translate([wheel_radius/2,0,0])rotate([90,0,0])cylinder(h=cylinder_height,r=cylinder_radius,cent
er=true);

// Cylinder 2

translate([0,0,wheel_radius/2])rotate([90,0,0])cylinder(h=cylinder_height,r=cylinder_radius,cent
er=true);

// Cylinder 3

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

// Cylinder 4

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

}

}
```



There are a few things that you need to get right. The first thing you should notice is that in order to define a module you have to type the word module followed by a name which you want to give to this module. In this case the module is named wheel. After the name of the module follows a pair of parentheses. Currently there is nothing inside the parentheses because no parameters have been defined for this module. Finally, after the pair of parentheses follows a pair of curly brackets. All commands that defined the corresponding object are placed inside the curly brackets. A semicolon is not required at the end.

The second thing you should notice is that OpenSCAD has not created any wheel. This is because you have just defined the wheel module but have not used it yet. In order to create a wheel you need to add a statement that creates a wheel, similar to how you would add a statement to create any primitive object (cube, sphere etc.).

module wheel() {

wheel_radius=10;

side_spheres_radius=50;

hub_thickness=4;

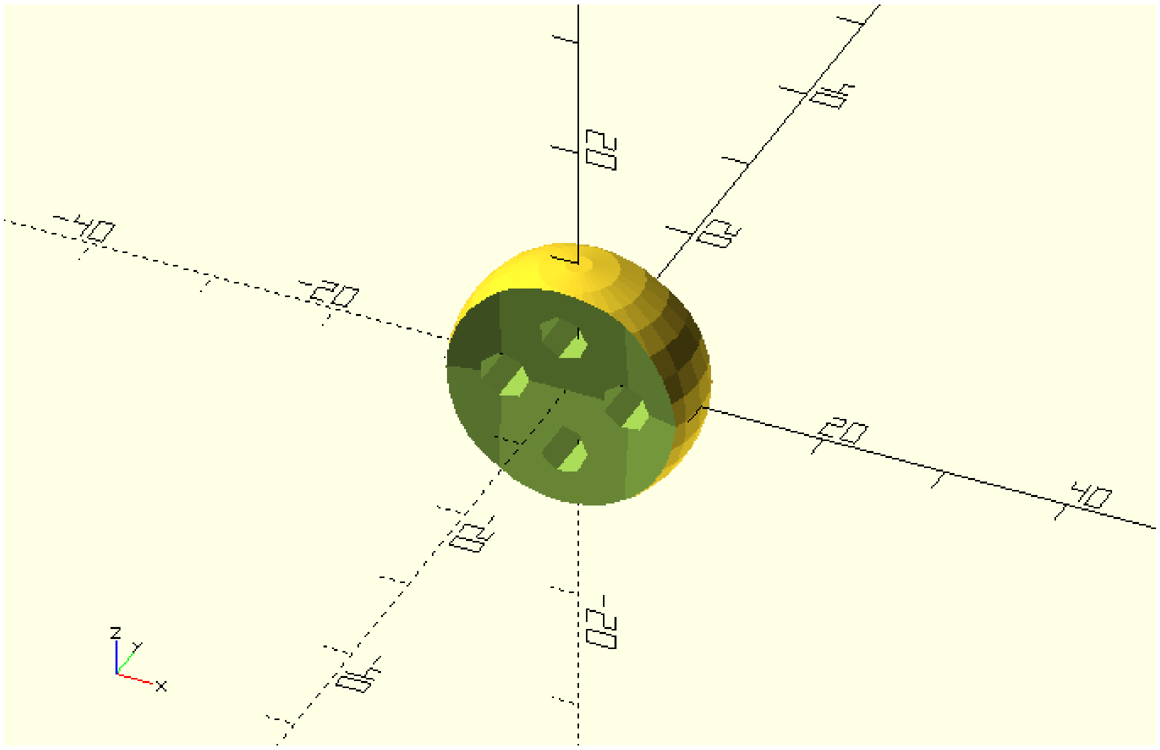
cylinder_radius=2;

cylinder_height=2*wheel_radius;

difference(){

// Wheel sphere

```
sphere(r=wheel_radius);

// Side sphere 1

translate([0,side_spheres_radius + hub_thickness/2,0])sphere(r=side_spheres_radius);

// Side sphere 2

translate([0,- (side_spheres_radius + hub_thickness/2),0])sphere(r=side_spheres_radius);

// Cylinder 1

translate([wheel_radius/2,0,0])rotate([90,0,0])cylinder(h=cylinder_height,r=cylinder_radius,cent
er=true);

// Cylinder 2

translate([0,0,wheel_radius/2])rotate([90,0,0])cylinder(h=cylinder_height,r=cylinder_radius,cent
er=true);

// Cylinder 3

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

// Cylinder 4

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

}

}

wheel();
```

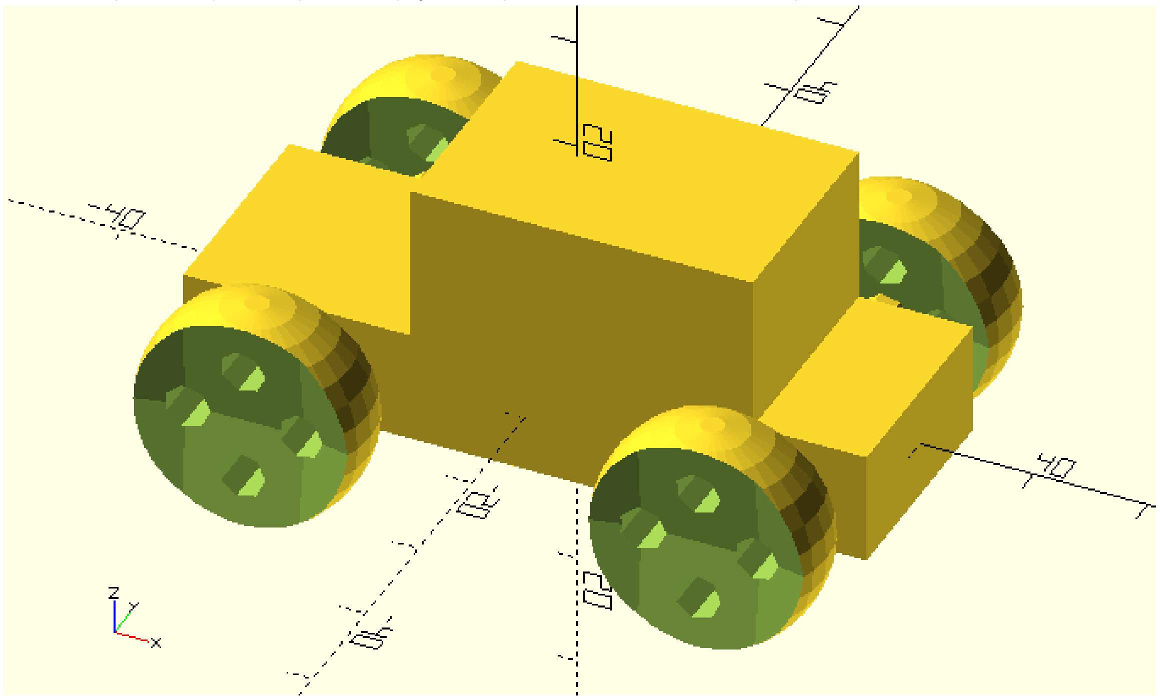


You can thing of defining modules as extending the OpenSCAD scripting language. When you have defined a wheel module it's like having an additional available primitive object. In this case the new object is the wheel that you have defined. You can then use this module similar to how you would use any other available primitive.

Try defining the above wheel module in the car's script. Try creating the wheels of the car using the defined wheel module.

```
module wheel() {

wheel_radius=10;

side_spheres_radius=50;

hub_thickness=4;

cylinder_radius=2;

cylinder_height=2*wheel_radius;

difference(){

// Wheel sphere

sphere(r=wheel_radius);

// Side sphere 1

translate([0,side_spheres_radius + hub_thickness/2,0])sphere(r=side_spheres_radius);
// Side sphere 2

translate([0,- (side_spheres_radius + hub_thickness/2),0])sphere(r=side_spheres_radius);

// Cylinder 1

translate([wheel_radius/2,0,0])rotate([90,0,0])cylinder(h=cylinder_height,r=cylinder_radius,cent
er=true);

// Cylinder 2

translate([0,0,wheel_radius/2])rotate([90,0,0])cylinder(h=cylinder_height,r=cylinder_radius,cent
er=true);

// Cylinder 3

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

// Cylinder 4

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

}

}


base_height = 10;

top_height = 14;

track = 35;

body_roll = 0;

wheels_turn = 0;


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
```

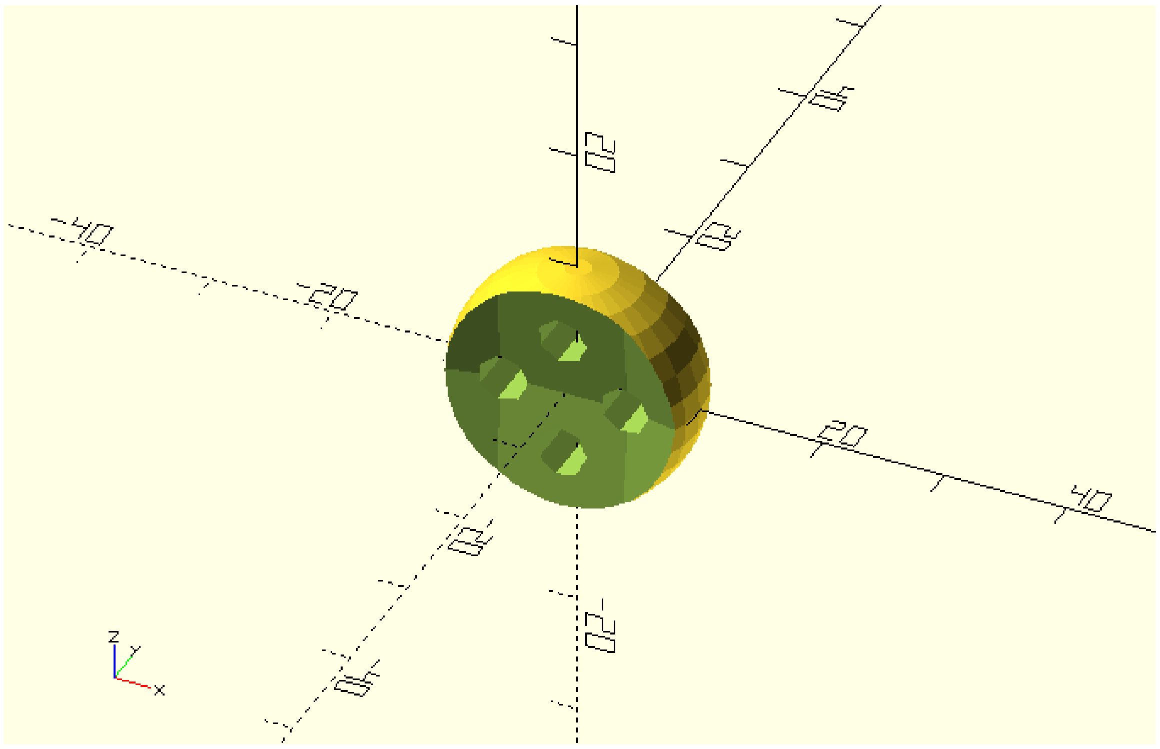
```
translate([-20,-track/2,0])rotate([0,0,wheels_turn])wheel();

 // Front right wheel

translate([-20,track/2,0])rotate([0,0,wheels_turn])wheel();

// Rear left wheel

translate([20,-track/2,0])rotate([0,0,0])wheel();

// Rear right wheel

translate([20,track/2,0])rotate([0,0,0])wheel();

// 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);
```



Parameterizing modules

The wheel design that was specified in the wheel module has a number of variables that can be used to customize it. These variables are defined inside the curly brackets of the wheel module's definition. As a result, while the output of the wheel module can be customized, the wheel module itself can create only one version of the wheel which corresponds to the values of the defined variables. This means the wheel module can't be used to create different wheels for the front and back axles. If you have been getting a feeling of the good practices of parametric design, you should realize that such a thing is not desired. It would be way batter if the wheel module could be used to create different versions of the wheel. For this to happen the variables that are defined and used inside the wheel module, need to be defined as parameters of the wheel module instead. This can be done in the following way.

```
module wheel(wheel_radius, side_spheres_radius, hub_thickness, cylinder_radius) {

cylinder_height=2*wheel_radius;

difference(){

// Wheel sphere

sphere(r=wheel_radius);

// Side sphere 1

translate([0,side_spheres_radius + hub_thickness/2,0])sphere(r=side_spheres_radius);

// Side sphere 2

translate([0,- (side_spheres_radius + hub_thickness/2),0])sphere(r=side_spheres_radius);

// Cylinder 1

translate([wheel_radius/2,0,0])rotate([90,0,0])cylinder(h=cylinder_height,r=cylinder_radius,center=true);

// Cylinder 2

translate([0,0,wheel_radius/2])rotate([90,0,0])cylinder(h=cylinder_height,r=cylinder_radius,center=true);

// Cylinder 3

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

// Cylinder 4

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

}

}

wheel(wheel_radius=10, side_spheres_radius=50, hub_thickness=4, cylinder_radius=2);
```

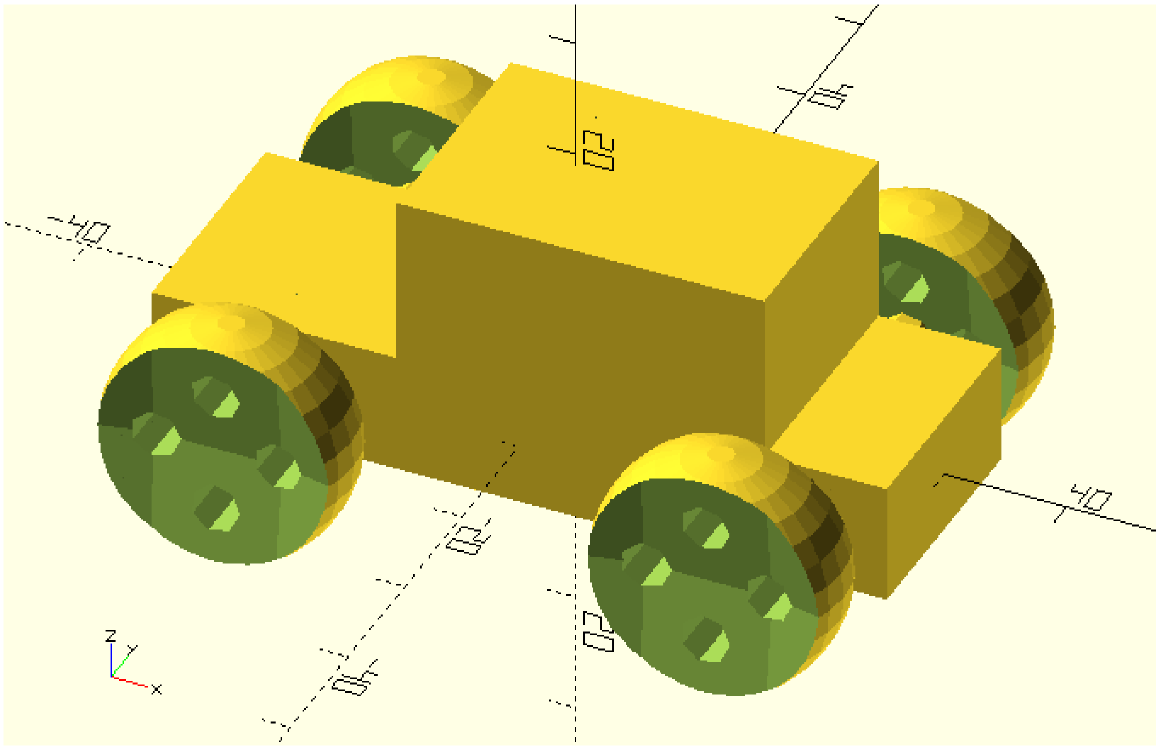


You should notice the definition of the module's parameters inside the parentheses. You should also notice that the value of each parameter is no longer assigned inside the curly brackets of the module's definitions. Instead, the value of the parameters is defined every time the modules is called. As a result, the module can now be used to create different versions of the wheel.

Try defining the above wheel module in the car's script. Try creating the car's wheels by using the wheel module. When calling the wheel module pass the values of 5, 50, 4 and 2 to the corresponding wheel_radius, side_spheres_radius, hub_thickness and cylinder_radius parameters.

```
module wheel(wheel_radius, side_spheres_radius, hub_thickness, cylinder_radius) {

cylinder_height=2*wheel_radius;

difference(){

// Wheel sphere

sphere(r=wheel_radius);

// Side sphere 1

translate([0,side_spheres_radius + hub_thickness/2,0])sphere(r=side_spheres_radius);

// Side sphere 2

translate([0,- (side_spheres_radius + hub_thickness/2),0])sphere(r=side_spheres_radius);

// Cylinder 1
translate([wheel_radius/2,0,0])rotate([90,0,0])cylinder(h=cylinder_height,r=cylinder_radius,cent
er=true);

// Cylinder 2

translate([0,0,wheel_radius/2])rotate([90,0,0])cylinder(h=cylinder_height,r=cylinder_radius,cent
er=true);

// Cylinder 3

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

// Cylinder 4

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

}

}



base_height = 10;

top_height = 14;

track = 35;

body_roll = 0;

wheels_turn = 0;


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([0,0,wheels_turn])wheel(wheel_radius=10,
side_spheres_radius=50, hub_thickness=4, cylinder_radius=2);
```

```
// Front right wheel

translate([-20,track/2,0])rotate([0,0,wheels_turn])wheel(wheel_radius=10,
side_spheres_radius=50, hub_thickness=4, cylinder_radius=2);

// Rear left wheel

translate([20,-track/2,0])rotate([0,0,0])wheel(wheel_radius=10, side_spheres_radius=50,
hub_thickness=4, cylinder_radius=2);

// Rear right wheel

translate([20,track/2,0])rotate([0,0,0])wheel(wheel_radius=10, side_spheres_radius=50,
hub_thickness=4, cylinder_radius=2);

// 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 defining a wheel_radius, side_spheres_radius, hub_thickness and cylinder_radius variable in the car's script and assign the values of 10, 50, 4 and 2 accordingly. Try using these variables to define the values of the wheel_radius, side_spheres_radius, hub_thickness and cylinder_radius parameters when calling the wheel module.

```
wheel_radius=10;
```

```
side_spheres_radius=50;

hub_thickness=4;

cylinder_radius=2;

wheel(wheel_radius=wheel_radius, side_spheres_radius=side_spheres_radius,
hub_thickness=hub_thickness, cylinder_radius=cylinder_radius);
```

Try defining different wheel_radius, side_spheres_radius, hub_thickness and cylinder_radius variables for the front and rear axles. Try assigning a combination of values that you like to these variables. Remember to also edit the name of the variables in the respective calls of the wheel module.

```
module wheel(wheel_radius, side_spheres_radius, hub_thickness, cylinder_radius) {

cylinder_height=2*wheel_radius;

difference(){

// Wheel sphere

sphere(r=wheel_radius);

// Side sphere 1

translate([0,side_spheres_radius + hub_thickness/2,0])sphere(r=side_spheres_radius);

// Side sphere 2

translate([0,- (side_spheres_radius + hub_thickness/2),0])sphere(r=side_spheres_radius);

// Cylinder 1

translate([wheel_radius/2,0,0])rotate([90,0,0])cylinder(h=cylinder_height,r=cylinder_radius,cent
er=true);

// Cylinder 2

translate([0,0,wheel_radius/2])rotate([90,0,0])cylinder(h=cylinder_height,r=cylinder_radius,cent
er=true);

// Cylinder 3

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

// Cylinder 4

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

}
}



base_height = 10;

top_height = 14;

track = 35;

body_roll = 0;

wheels_turn = 0;


wheel_radius_front=10;

side_spheres_radius_front=50;

hub_thickness_front=4;

cylinder_radius_front=2;


wheel_radius_rear=12;

side_spheres_radius_rear=30;

hub_thickness_rear=8;

cylinder_radius_rear=3;


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([0,0,wheels_turn])wheel(wheel_radius=wheel_radius_front,
side_spheres_radius=side_spheres_radius_front, hub_thickness=hub_thickness_front,
cylinder_radius=cylinder_radius_front);
```

// Front right wheel

translate([-20,track/2,0])rotate([0,0,wheels_turn])wheel(wheel_radius=wheel_radius_front, side_spheres_radius=side_spheres_radius_front, hub_thickness=hub_thickness_front, cylinder_radius=cylinder_radius_front);

// Rear left wheel

translate([20,-track/2,0])rotate([0,0,0])wheel(wheel_radius=wheel_radius_rear, side_spheres_radius=side_spheres_radius_rear, hub_thickness=hub_thickness_rear, cylinder_radius=cylinder_radius_rear);

// Rear right wheel

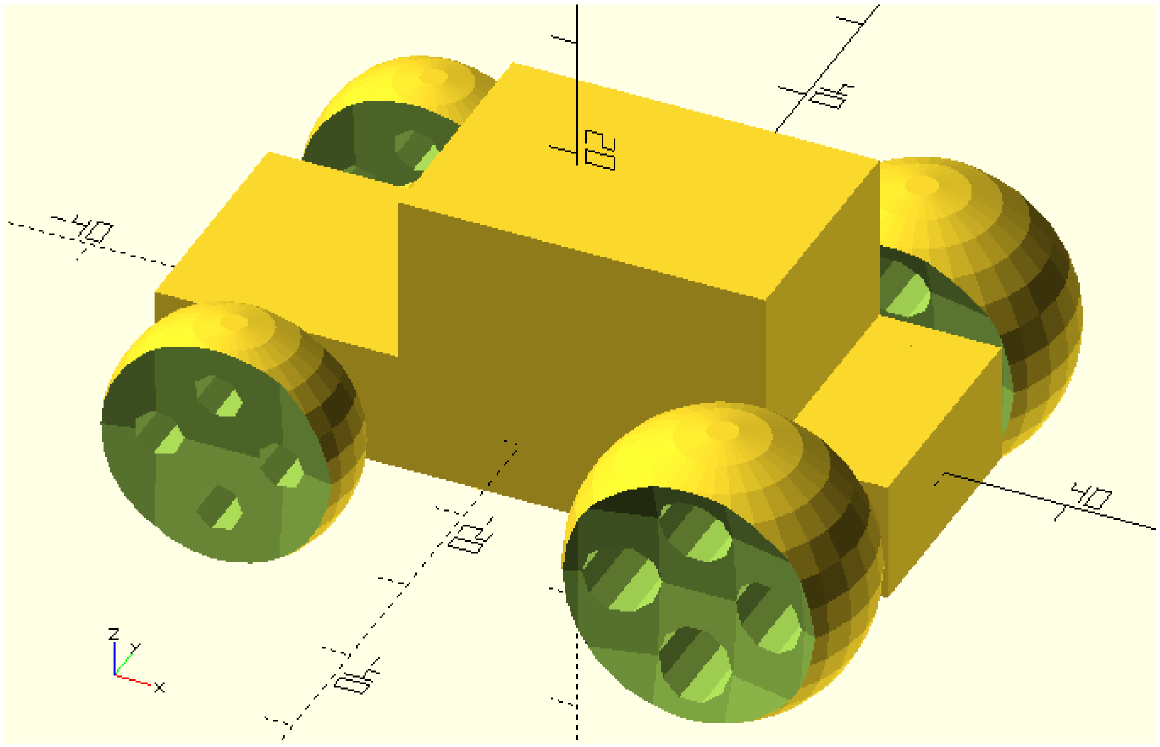
translate([20,track/2,0])rotate([0,0,0])wheel(wheel_radius=wheel_radius_rear, side_spheres_radius=side_spheres_radius_rear, hub_thickness=hub_thickness_rear, cylinder_radius=cylinder_radius_rear);

// 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);



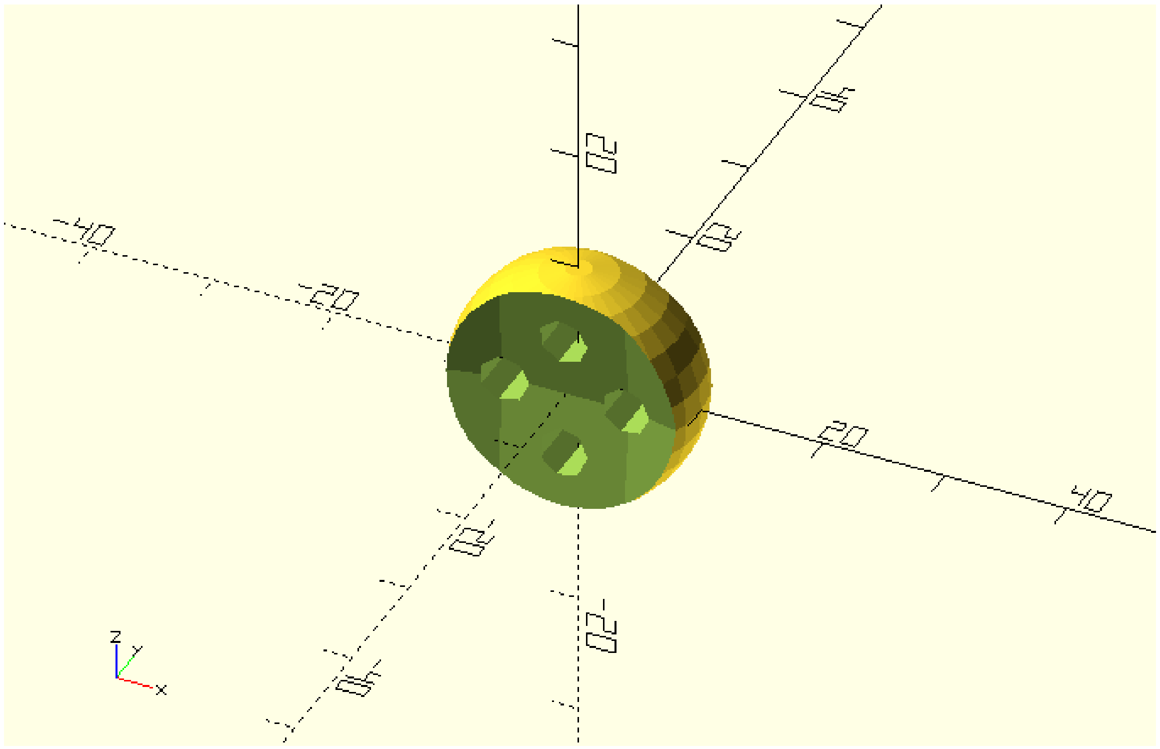
Defining default values of module's parameters

You can set a specific combination of values for the wheel module's parameters as default. This can be achieved in the following way.

```
module wheel(wheel_radius=10, side_spheres_radius=50, hub_thickness=4, cylinder_radius=2) {

cylinder_height=2*wheel_radius;

difference(){

// Wheel sphere

sphere(r=wheel_radius);

// Side sphere 1

translate([0,side_spheres_radius + hub_thickness/2,0])sphere(r=side_spheres_radius);

// Side sphere 2

translate([0,- (side_spheres_radius + hub_thickness/2),0])sphere(r=side_spheres_radius);

// Cylinder 1

translate([wheel_radius/2,0,0])rotate([90,0,0])cylinder(h=cylinder_height,r=cylinder_radius,cent
er=true);

// Cylinder 2

translate([0,0,wheel_radius/2])rotate([90,0,0])cylinder(h=cylinder_height,r=cylinder_radius,cent
er=true);

// Cylinder 3

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

// Cylinder 4

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

}

}
```

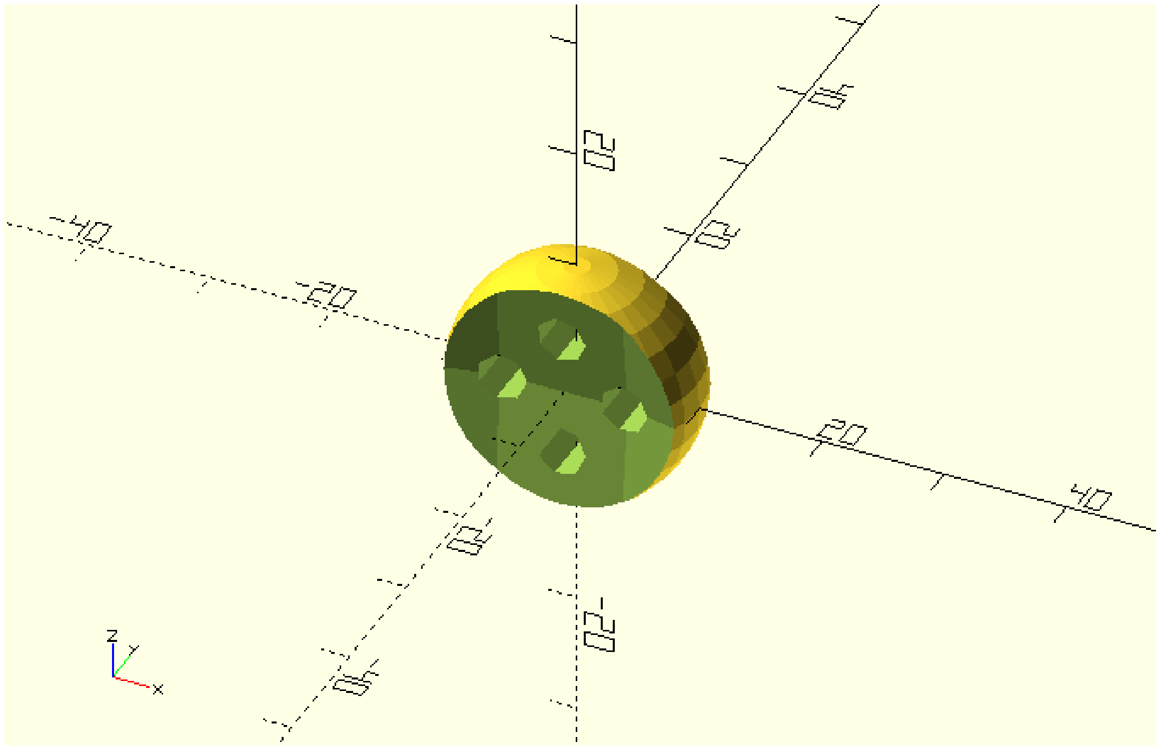
You should notice that the default values are assigned inside the parentheses at the definition of the module. By defining default values for the module's parameters, you have more flexibility in the way the wheel module is used. For example, the simplest way to use the module is without specifying any parameters when calling it.

```
wheel();
```

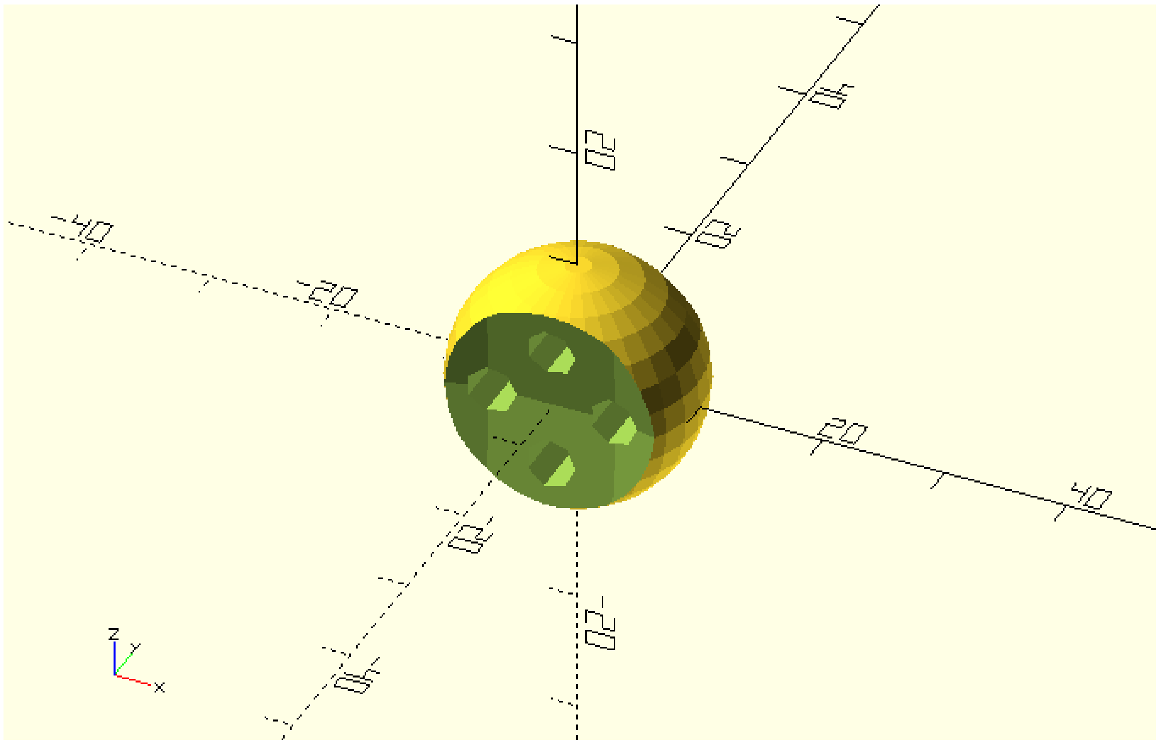


If a parameter’s value is not specified when calling the wheel module, the default value for this parameter is used. The default values can be set equal to the most used version of the wheel. The default values can be overridden by assigning a new value to the corresponding parameters when calling the wheel module. None or any number of default values may be overridden. Thus, by specifying default values the wheel module can be used in any of the following ways as well as in many more.

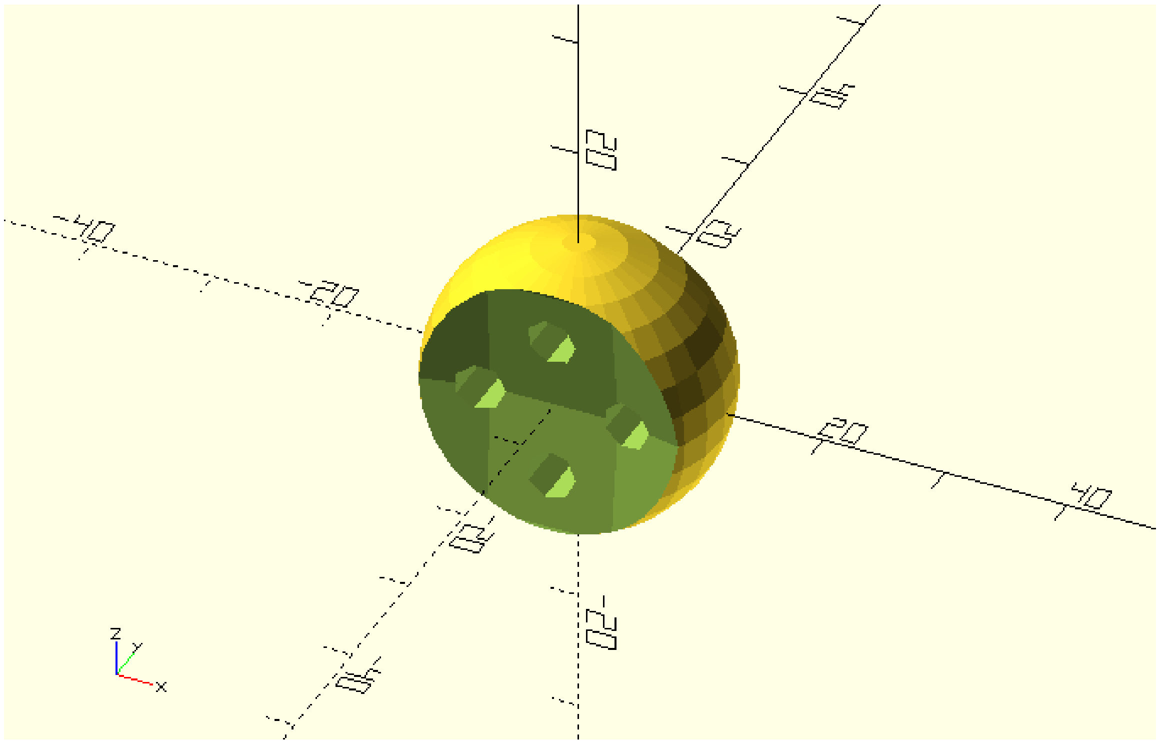
```
wheel();
```



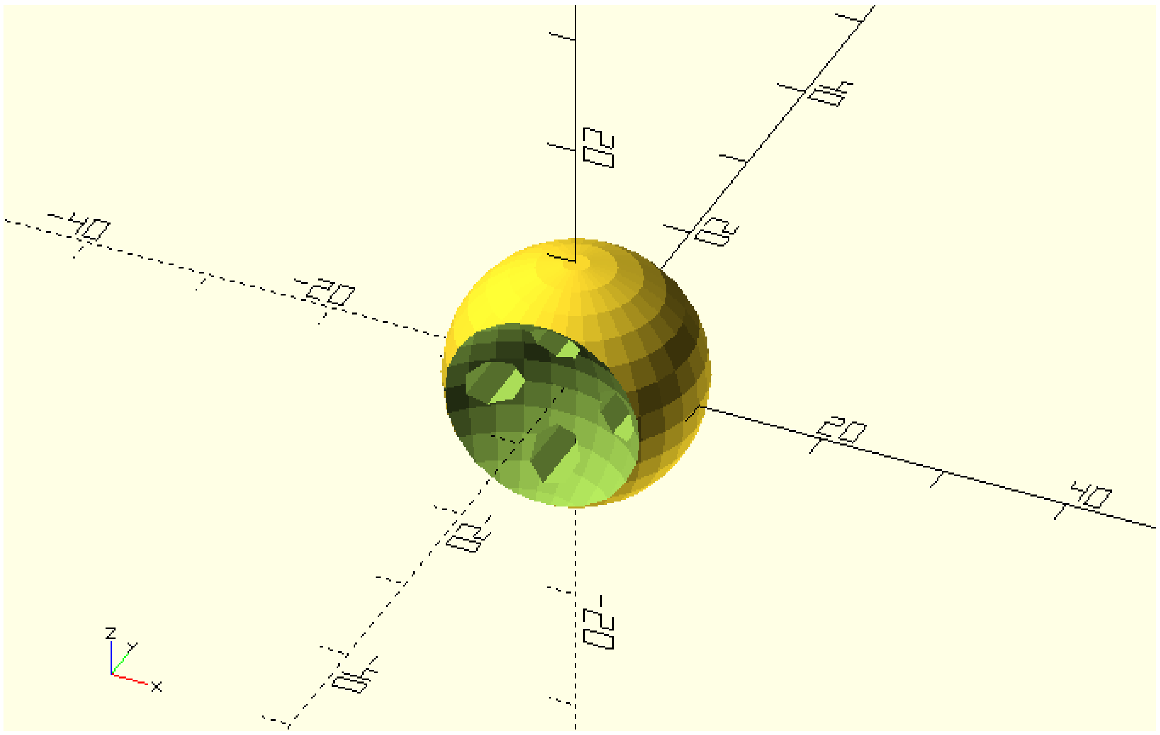
wheel(hub_thickness=8);



wheel(hub_thickness=8, wheel_radius=12);



Include default values in the definition of the wheel module. Try creating a few wheels by overriding any number of default values. Can you make a wheel that looks like the following?



```
wheel(side_spheres_radius=10);
```

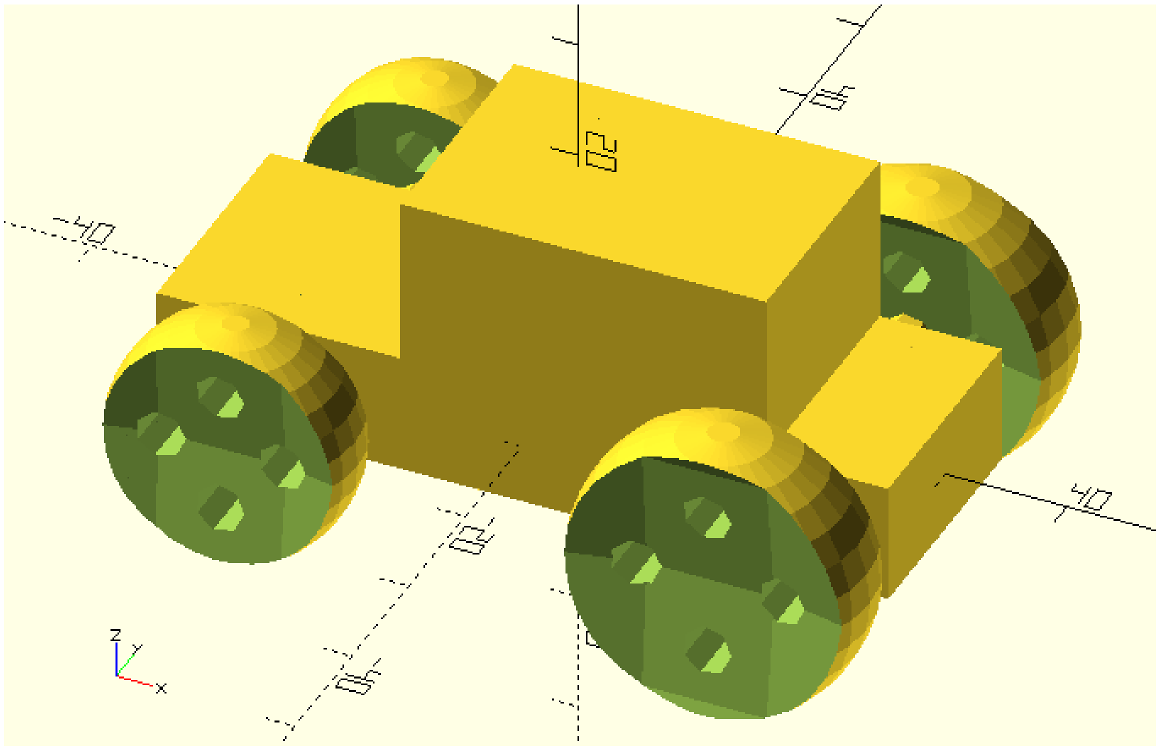
Separating the whole model into modules

The use of modules is a very powerful feature of OpenSCAD. You should start thinking of your models as a combination of modules. For example, the car model can be thought of as a combination of a body, wheel and axle module. This opens the possibilities of further reusing and recombining your modules to create different models.

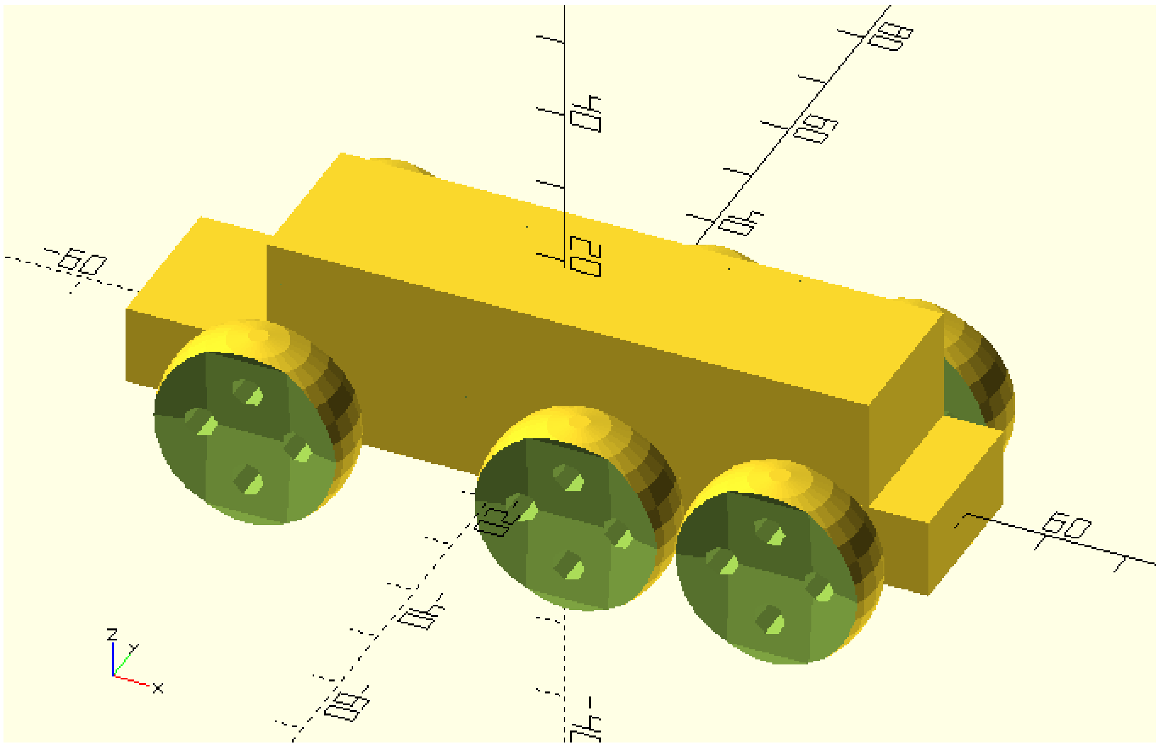
Try defining a body and an axle module. What parameters should the body and axle modules have? Try recreating the car using the body, wheel and axle modules. Give the parameters of the wheel module a default set of values that corresponds to the front wheels. Pass different values to the wheel module when creating the rear wheels by defining appropriate variables in your script. Set default values for the parameters of the body and axle modules too.

```
module wheel(wheel_radius=10, side_spheres_radius=50, hub_thickness=4, cylinder_radius=2) {

cylinder_height=2*wheel_radius;

difference(){

// Wheel sphere

sphere(r=wheel_radius);

// Side sphere 1

translate([0,side_spheres_radius + hub_thickness/2,0])sphere(r=side_spheres_radius);

// Side sphere 2

translate([0,- (side_spheres_radius + hub_thickness/2),0])sphere(r=side_spheres_radius);

// Cylinder 1

translate([wheel_radius/2,0,0])rotate([90,0,0])cylinder(h=cylinder_height,r=cylinder_radius,center=true);

// Cylinder 2

translate([0,0,wheel_radius/2])rotate([90,0,0])cylinder(h=cylinder_height,r=cylinder_radius,center=true);

// Cylinder 3

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

// Cylinder 4

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

}
}


module body(base_height=10, top_height=14, base_length=60, top_length=30, width=20,
top_offset=5) {
   // Car body base
   cube([base_length,width,base_height],center=true);
   // Car body top
translate([top_offset,0,base_height/2+top_height/2])cube([top_length,width,top_height],cente
r=true);
}


module axle(track=35, radius=2) {
   rotate([90,0,0])cylinder(h=track,r=2,center=true);
}


wheelbase = 40;
track = 35;
body_roll = 0;
wheels_turn = 0;

wheel_radius_rear=12;

// Body
rotate([body_roll,0,0]){
   body();
}
// Front left wheel
translate([-wheelbase/2,-track/2,0])rotate([0,0,wheels_turn])wheel();
 // Front right wheel
```

```
translate([-wheelbase/2,track/2,0])rotate([0,0,wheels_turn])wheel();

// Rear left wheel

translate([wheelbase/2,-track/2,0])rotate([0,0,0])wheel(wheel_radius=wheel_radius_rear);

// Rear right wheel

translate([wheelbase/2,track/2,0])rotate([0,0,0])wheel(wheel_radius=wheel_radius_rear);

// Front axle

translate([-wheelbase/2,0,0])axle();

// Rear axle

translate([wheelbase/2,0,0])axle();
```



Try reusing the body, wheel and axle modules to create vehicle that looks similar to the following.



```
module wheel(wheel_radius=10, side_spheres_radius=50, hub_thickness=4, cylinder_radius=2) {

cylinder_height=2*wheel_radius;

difference(){

// Wheel sphere

sphere(r=wheel_radius);

// Side sphere 1

translate([0,side_spheres_radius + hub_thickness/2,0])sphere(r=side_spheres_radius);

// Side sphere 2

translate([0,- (side_spheres_radius + hub_thickness/2),0])sphere(r=side_spheres_radius);

// Cylinder 1

translate([wheel_radius/2,0,0])rotate([90,0,0])cylinder(h=cylinder_height,r=cylinder_radius,center=true);

// Cylinder 2

translate([0,0,wheel_radius/2])rotate([90,0,0])cylinder(h=cylinder_height,r=cylinder_radius,center=true);

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

// Cylinder 4

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

}

}


module body(base_height=10, top_height=14, base_length=60, top_length=30, width=20,
top_offset=5) {

   // Car body base

   cube([base_length,width,base_height],center=true);

   // Car body top

translate([top_offset,0,base_height/2+top_height/2])cube([top_length,width,top_height],cente
r=true);

}


module axle(track=35, radius=2) {

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

}


track = 35;

body_roll = 0;

wheels_turn = 0;


base_length = 100;

top_length = 75;

top_offset = 5;


front_axle_offset = 30;
```

```
rear_axle_1_offset = 10;

rear_axle_2_offset = 35;


wheel_radius = 12;


// Body

rotate([body_roll,0,0]){

   body(base_length=base_length, top_length=top_length, top_offset=top_offset);

}

// Front left wheel

translate([-front_axle_offset,-
track/2,0])rotate([0,0,wheels_turn])wheel(wheel_radius=wheel_radius);

 // Front right wheel

translate([-
front_axle_offset,track/2,0])rotate([0,0,wheels_turn])wheel(wheel_radius=wheel_radius);

// Rear left wheel 1

translate([rear_axle_1_offset,-track/2,0])rotate([0,0,0])wheel(wheel_radius=wheel_radius);

// Rear right wheel 1

translate([rear_axle_1_offset,track/2,0])rotate([0,0,0])wheel(wheel_radius=wheel_radius);

// Rear left wheel 2

translate([rear_axle_2_offset,-track/2,0])rotate([0,0,0])wheel(wheel_radius=wheel_radius);

// Rear right wheel 2

translate([rear_axle_2_offset,track/2,0])rotate([0,0,0])wheel(wheel_radius=wheel_radius);

// Front axle

translate([-front_axle_offset,0,0])axle();

// Rear axle 1

translate([rear_axle_1_offset,0,0])axle();

// Rear axle 2

translate([rear_axle_2_offset,0,0])axle();
```