translate ([X-Achse, Y-Achse, Z-Achse])

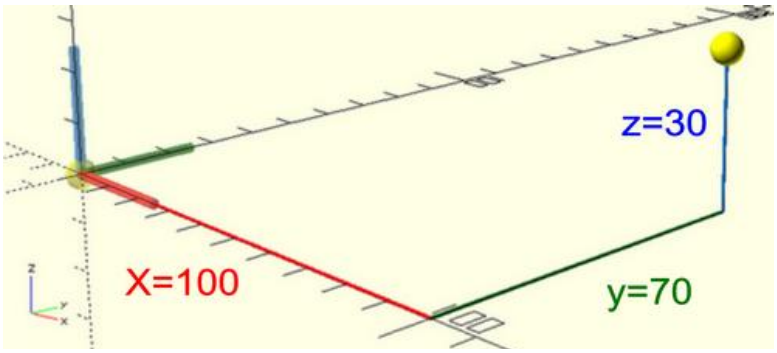
Formen an einen bestimmten Ort verschieben

Mit dem Befehl "translate" wird das Element im Raum versetzt. Über die drei Angaben der Achsen X, Y und Z wird es punktgenau neu an Ort und Stelle verschoben.

translate ([**x** , **y** , **z**])

Es folgt kein Semikolon. Diese Aufgabe erfüllt der nachfolgende Befehl zur Erzeugung des Elementes.

Der Versatzbefehl "translate" wird dem jeweiligen Objekt vorangestellt. Im Bild sind die drei Achsen mit den jeweiligen Farben versehen:



Die folgende Anweisung translate([**100**, **70**, **30**]) bestimmt nun den versetzten Startpunkt um von dort aus die neue Kugel zu zeichnen.

Die Standardposition wird somit um 100 Einheiten in die Richtung entlang der X-Achse, 70 Einheiten in positiver Richtung entlang der Y-Achse und 30 Einheiten entlang der Z-Achse verschoben:

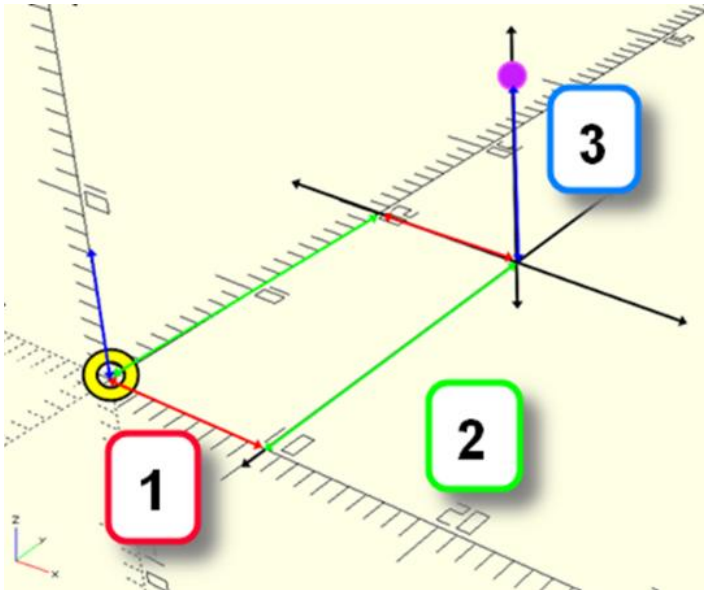
```
translate([100, 70, 30]) sphere(3);
```

Der „translate“-Befehl verwendet zusätzlich eckige Klammern, um X, Y und Z als Parameterangaben zu gruppieren. Ähnlich wie bei der Angabe der Abmessungen eines Quaders.

Wichtig ist die Reihenfolge der Zahlen! Die erste Zahl beschreibt die Bewegung entlang der X-Achse; die zweite entlang der Y-Achse; und die dritte entlang der Z-Achse.
„translate“ hat **kein Semikolon**, den sonst würde gleich danach der Befehl zu Ende sein, bevor das folgende Objekt gezeichnet worden wäre.
Beendet wird die gesamte Anweisung mit dem Semikolon **nach** dem zu zeichnenden Objekt.

003-02

Beispiel:



translate([10, 20, 7])
soll unsere Veränderung des Objektes werden

Der Nullpunkt (der gelbe Ring im Nullpunkt) wird auf der X-Achse um 10 verschoben, auf der Y-Achse um 20 und auf der Z-Achse um 7.

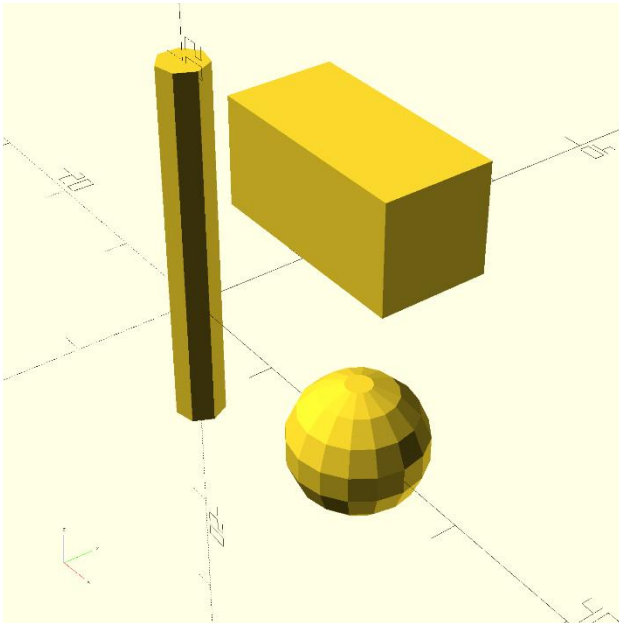
Somit steht der neue Startpunkt (lila Punkt) für das Objekt fest.

Um ein komplexeres Design zu erstellen, müssen möglicherweise Formen in verschiedene Positionen verschoben werden.
Es wird einfach „translate“ mit Parametern (ohne Semikolon) vor einen anderen Befehl gesetzt.

```
translate([-10, 10, 0]) cube([20, 10, 10]);
translate([20, 0, 0]) sphere(5);
translate([0, 0, -10]) cylinder(h=30, r1=2, r2=2);
```

Im Bild:

Drei unterschiedliche Formen, verschoben aus deren Standardpositionen



Sowohl die Kugel als auch der Zylinder bewegen sich gemäß ihrem jeweiligen Mittelpunkt, während sich der Würfel relativ zu der Ecke bewegt, die den Nullpunkt berührt.

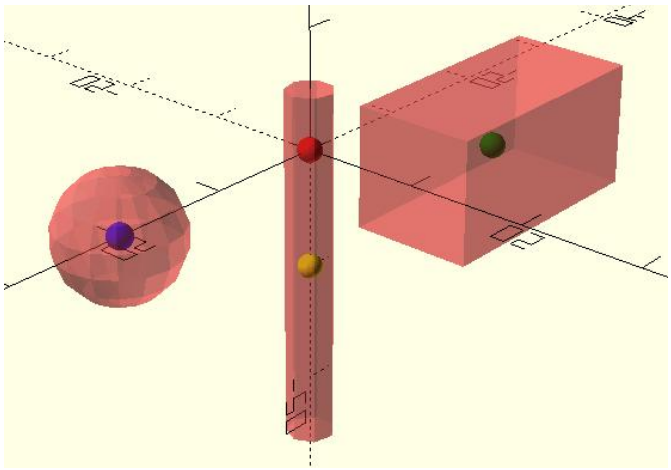
Die Verschiebung bringt ein ganz anderes Ergebnis zutage, wenn dieselbe Verschiebung an einem Würfel und Zylinder ausgeführt wird, die mittels „center=true“ ergänzt wurden.

```
translate([-10, 10, 0]) cube([20, 10, 10], center=true);
translate([20, 0, 0]) sphere(5);
translate([0, 0, -10]) cylinder(h=30, r1=2, r2=2, center=true);
```

Im Bild:

Drei unterschiedliche Formen, aus der jeweiligen zentrierten Position vom Nullpunkt aus wegbewegt.

Der rote Punkt ist wie immer der Nullpunkt. Davon ausgehend bewegen sich die drei Elemente mit ihren eigenen Mittelpunkten weg.



```
#translate([-10, 10, 0]) cube([20, 10, 10], center=true);
#translate([20, 0, 0]) sphere(5);
#translate([0, 0, -10]) cylinder(h=30, r1=2, r2=2, center=true);
color("Red") sphere(1,$fn=50);
translate([-10, 10, 0]) color("Green") sphere(1,$fn=50);
translate([20, 0, 0]) color("Blue") sphere(1,$fn=50);
translate([0, 0, -10]) color("Yellow") sphere(1,$fn=50);
```

003-04

rotate ([x, y, z]) Drehen von Formen

Normalerweise zeichnet OpenSCAD Formen so, dass sie auf eine bestimmte Weise ausgerichtet sind. Es zeichnet beispielsweise Kugeln mit dem Mittelpunkt (0, 0, 0) und Quader mit einer einzelnen Ecke bei (0, 0, 0). Manchmal möchten man diese jedoch anders ausrichten, in eine andere Richtung drehen.

Eine Möglichkeit die Standardposition einer Form zu ändern, besteht darin, sie zu drehen. Wird eine Form gedreht, wird nur der Rotationswinkel um jede der drei Achsen angegeben.

Mit dem Befehl "rotate" wird das Element im Raum gedreht. Über die drei Angaben der Achsen X, Y und Z wird es winkelgenau gedreht.

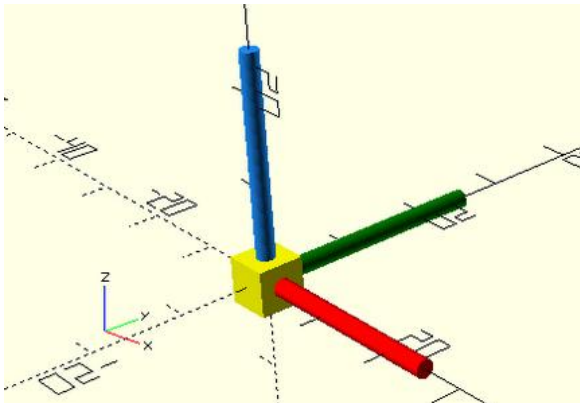
rotate ([**x** , **y** , **z**])

Es folgt kein Semikolon. Dies wird bei der Erstellung des nachfolgenden

Beispiel:
Mit einem Würfel im Nullpunkt (center=true) und den drei farbigen Achsen wird nun gedreht.

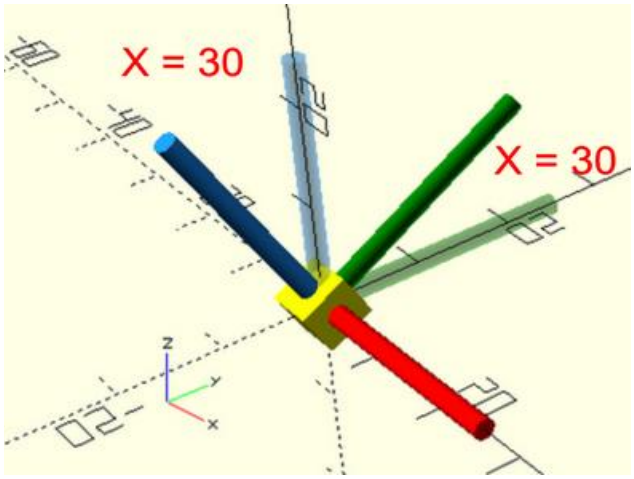
```
rotate([0, 0 , 0]) cube(5, center=true);
```

Alle Eingabewerte sind Winkel in Gradzahlen. Damit wird der Würfel noch nicht gedreht, da alle Werte noch mit Null versehen sind.



Nun wird alles um die X-Achse um 30 Grad positiv gedreht.

```
rotate([30, 0 , 0]) cube(5, center=true);
```

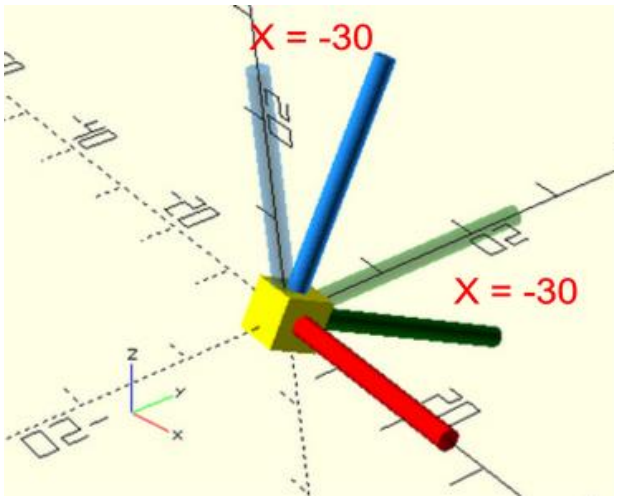


Sieht man auf der X-Achse zum Nullpunkt, so dreht sich durch den Wert Z=30 nun alles um 30 Grad **gegen den Uhrzeigersinn**.

Die ursprüngliche Position der Achsen wird durch blasse Farben dargestellt.

Dafür wird jetzt alles um die X-Achse um Minus 30 Grad gedreht:

```
rotate([- 30, 0 , 0]) cube(5, center=true);
```



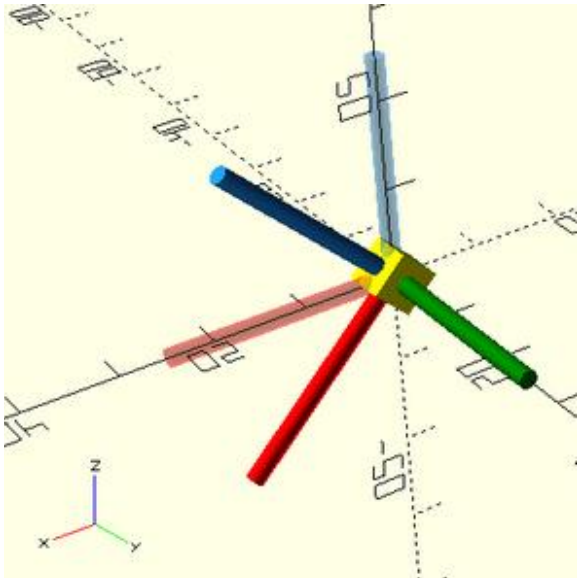
Sieht man auf der X-Achse zum Nullpunkt, so dreht sich durch den Wert Z=-30 jetzt alles um 30 Grad **im Uhrzeigersinn**.

Die ursprüngliche Position der Achsen wird durch blasse Farben dargestellt.

003-06

Drehen wir alles um die grüne Y-Achse um 45 Grad.

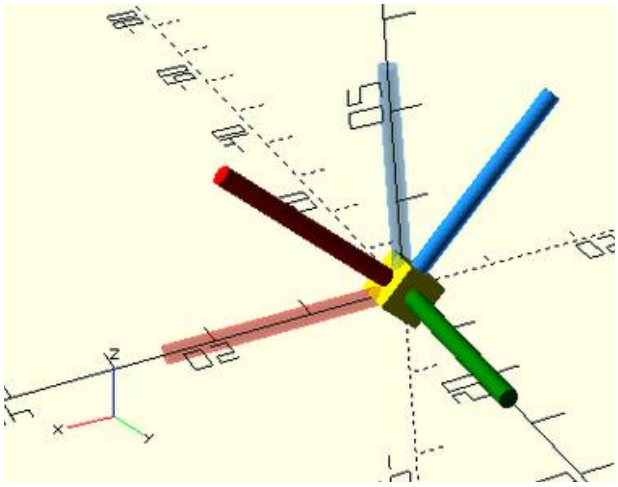
```
rotate([0, 45 , 0]) cube(5, center=true);
```



Sieht man auf der Y-Achse zum Nullpunkt, so dreht sich durch den Wert Y=45 nun alles um 45 Grad
gegen den Uhrzeigersinn.

Und jetzt wird alles um die grüne Y-Achse um Minus 45 Grad gedreht

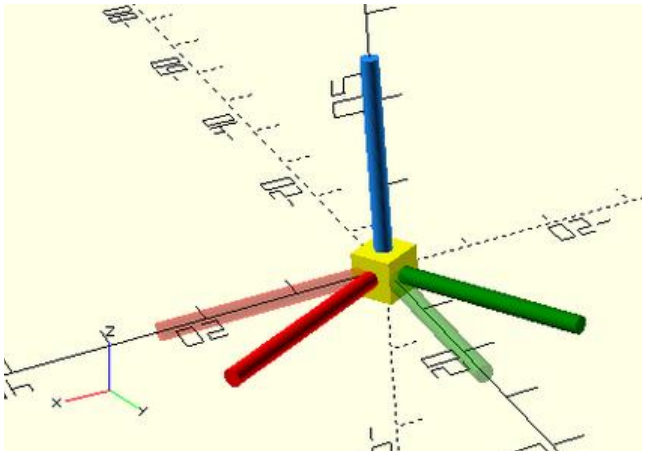
```
rotate([ 0, - 45, 0]) cube(5, center=true);
```



Sieht man auf der Y-Achse zum Nullpunkt, so dreht sich durch den Wert Y=-45 jetzt alles um 45 Grad
im Uhrzeigersinn.

Nicht anders bei der blauen Z-Achse 30 Grad.

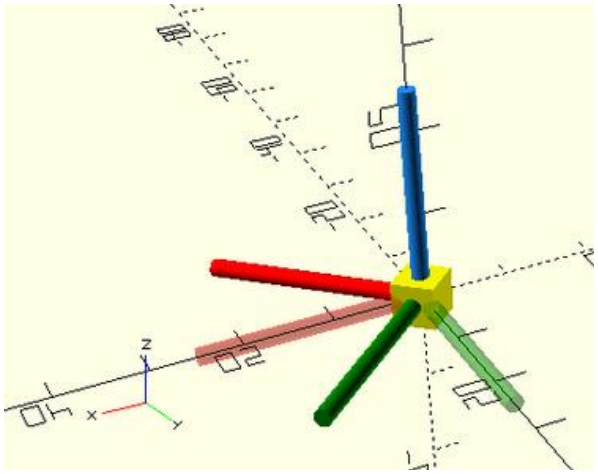
```
rotate([0, 0 , 30]) cube(5, center=true);
```



Sieht man auf der Z-Achse zum Nullpunkt, so dreht sich durch den Wert Z=30 nun alles um 30 Grad **gegen den Uhrzeigersinn**.

Ebenso bei Minus 45 Grad der blauen Achse.

```
rotate([ 0, 0, - 45 ]) cube(5, center=true);
```

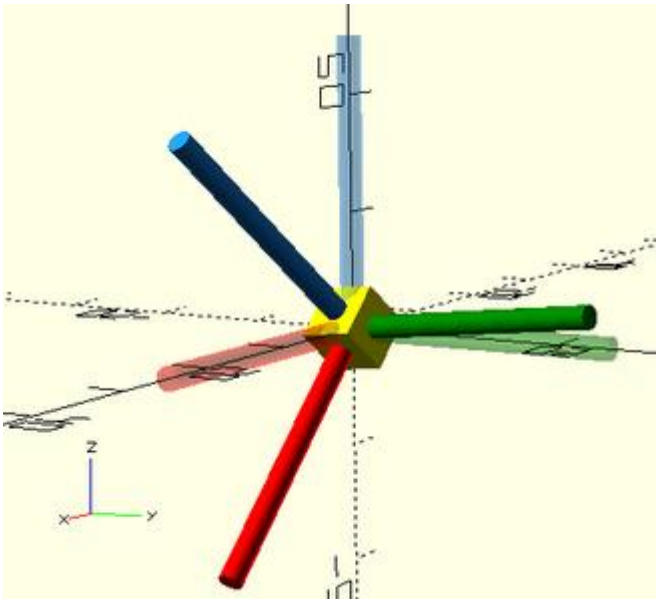


Sieht man auf der Z-Achse zum Nullpunkt, so dreht sich durch den Wert Z=-45 jetzt alles um 45 Grad **im Uhrzeigersinn**.

003-08

Kombination beim Drehen:

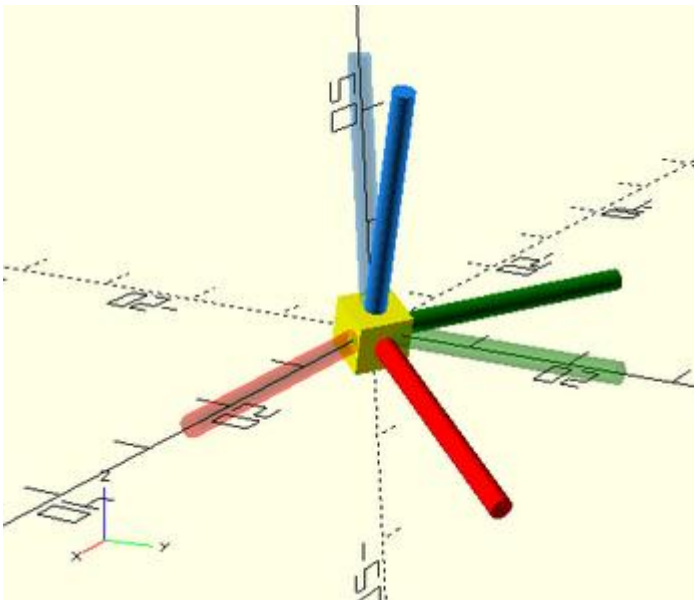
```
rotate([10, 45 , 0]) cube(5, center=true);
```



Hier wird nun:

auf der X-Achse um 10 Grad und auf der Y-Achse um 45 Grad jeweils gegen den Uhrzeigersinn gedreht.

```
rotate([ 0, 20, 60 ]) cube(5, center=true);
```



bei diesem Beispiel wird

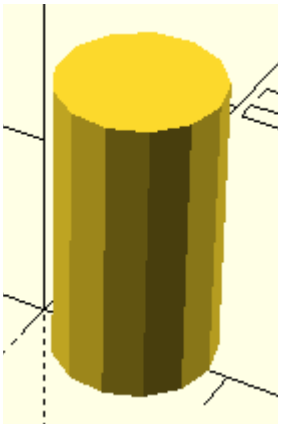
auf der Y-Achse um 20 Grad und auf der Z-Achse um 60 Grad jeweils gegen den Uhrzeigersinn gedreht.

Transformationen kombinieren

Es können Transformationen kombiniert werden, indem ein Befehl nach dem anderen aufgeführt wird.

Nehmen wir einen Zylinder als Grundform:

```
translate([5, 0, 0]) rotate([0, 0, 0]) cylinder(h=15, r=4);
```

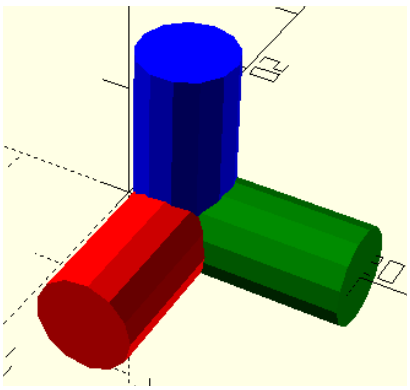


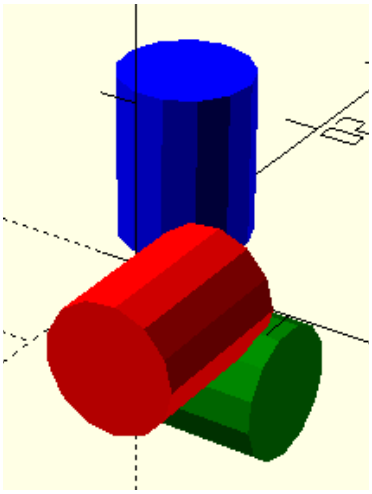
Ein Zylinder, der um 5 Einheiten versetzt wurde,
jedoch noch nicht gedreht.

Der folgende Code-Schnipsel wendet beispielsweise an jedem der drei Zylinder eine Verschiebung und anschließend eine Drehung an:

```
translate([5, 0, 0]) rotate([90, 0, 0]) cylinder(h=10, r=4);
translate([5, 0, 0]) rotate([0, 90, 0]) cylinder(h=10, r=4);
translate([5, 0, 0]) rotate([0, 0, 90]) cylinder(h=10, r=4);
```

Hier wurde jeder Zylinder mit der zu drehenden Achsenfarbe dargestellt.





Wird die Transformationen in umgekehrter Reihenfolge angewendet, erhält man ein anderes Ergebnis:

```
rotate([90, 0, 0]) translate([5, 0, 0]) cylinder(h=10, r=4);
rotate([0, 90, 0]) translate([5, 0, 0]) cylinder(h=10, r=4);
rotate([0, 0, 90]) translate([5, 0, 0]) cylinder(h=10, r=4);
```

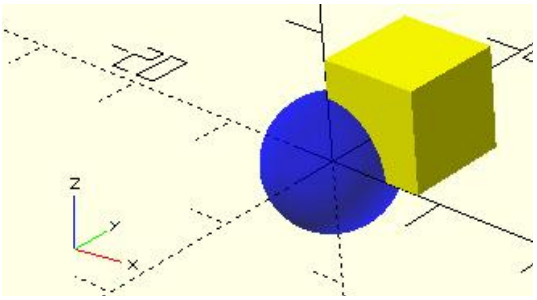
003-10

Beispiel 2: translate und rotate kombinieren - aber wie?

Zunächst wird ein Würfel und eine Kugel erstellt. Alles ohne jegliche Transformation.

```
// kombinationen
$fn=50;

translate([0,0,0])
rotate([0,0,0])
union(){
   color("Blue") sphere(5);
   color("Yellow")cube(8);
};
```

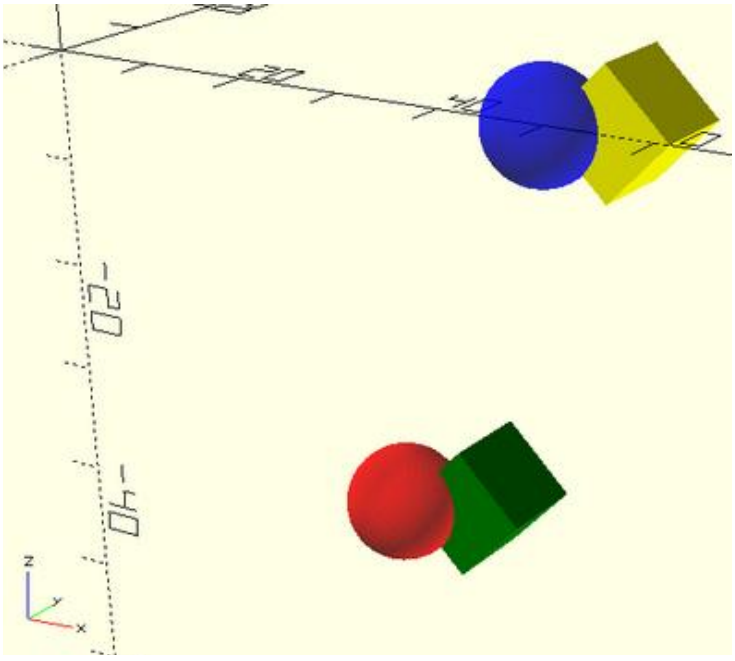


Dieser Script ergibt den gelben Würfel mit der blauen Kugel.

```
// kombinationen 13
$fn=50;

translate([50,0,0])
rotate([0,45,0])
union(){
   color("Blue",.6) sphere(5);
   color("Yellow")cube(8);
 };

rotate([0,45,0])
translate([50,0,0])
union(){
   color("Red",.6) sphere(5);
   color("Green")cube(8);
 };
```

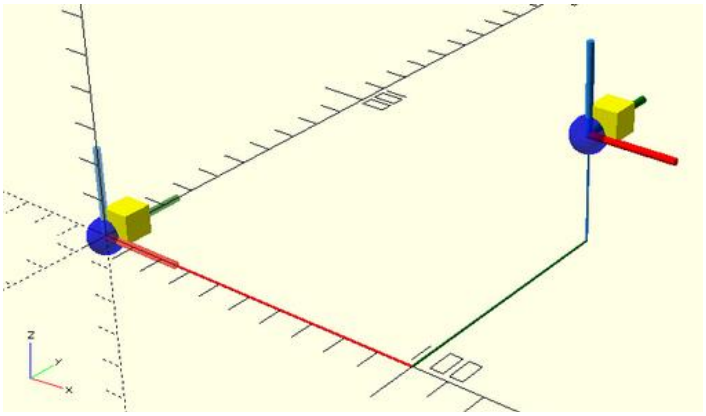


Bis auf die Darstellung der Farben der Elemente und die Reihenfolge von rotate und translate ist alles gleich …..

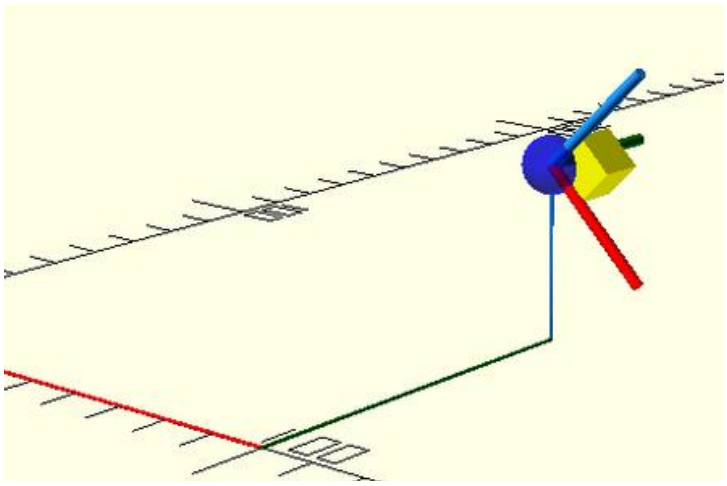
Was passiert bei: translate([50,0,0])rotate([0,45,0])...

1.)

"translate([100, 70, 30])..."
dies lässt zuerst beide Objekte (gelb und blau) auf der X, Y und Z-Achse versetzen.



2.)



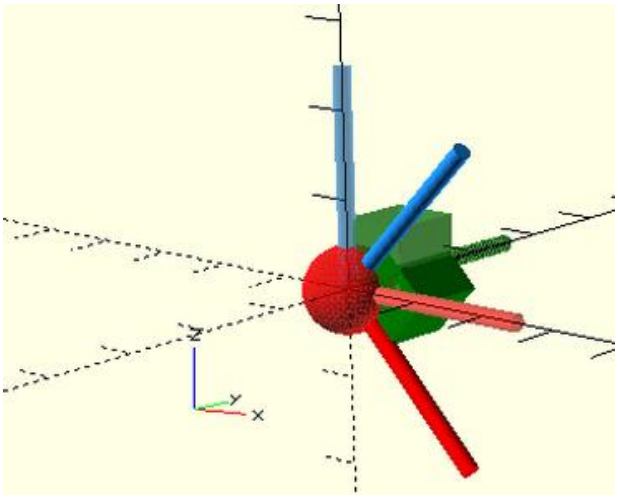
Danach schwenkt "... rotate([0,45,0]) ..."
um 45 Grad um die Y-Achse gegen den Uhrzeigersinn.

003-12

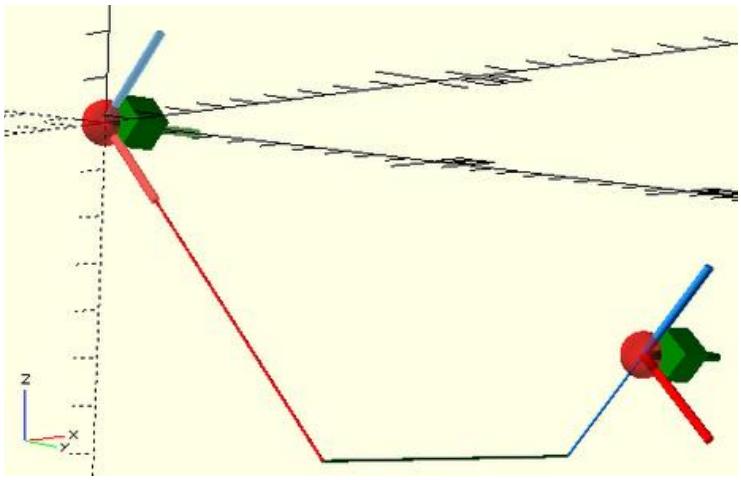
Was passiert bei: rotate([0,45,0])translate([50,0,0])...

1.)

"rotate([0,45,0])..." dies lässt zuerst beide Objekte(rot und grün) auf der Y-Achse um 45 Grad drehen.



2.)



"translate([100, 70, 30]" versetzt anschließend beide Objekte(rot und grün) auf den neuen X, Y und Z-Achsen.

Da beide Varianten "rotate und translate" oder "translate und rotate" möglich sind, ist es nun jedem selbst überlassen welche Kombination gewählt wird. Persönlich bevorzuge ich "translate und rotate". Irgendwie habe ich mich bereits daran gewöhnt: zuerst versetzen, dann drehen.

Als Muster soll dienen:

translate([0, 0, 0]) rotate([0, 0, 0]) cube([5, 5, 5]);
translate([15, 0, 0]) rotate([0, 0, 0]) cylinder(h=7, r=3);

So kann jetzt die Textzeile kopiert und in den Script eingefügt werden. Die Zahlen zu ändern dürfte kein Problem darstellen. Was nicht benötigt wird, einfach weglassen!