## mirror ( [ x, y, z ] ) **gespiegelte Formen**

Eine weitere Möglichkeit die Standardposition einer Form zu ändern, besteht darin, sie mit dem Befehl „mirror“ über eine imaginäre 2D-Ebene zu spiegeln.

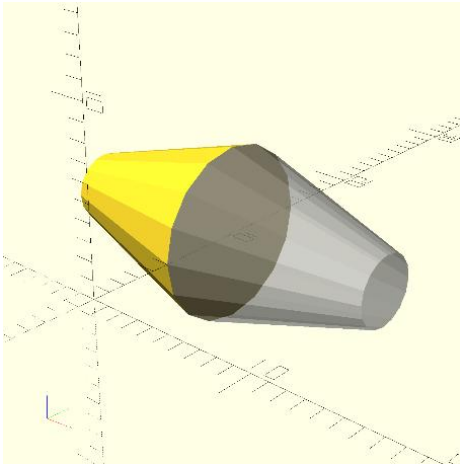
Wie der Name bereits vermuten lässt, erzeugt der Befehl eine Spiegelung der angegeben Form.

Das folgende Beispiel spiegelt einen Kegelstumpf über der YZ-Ebene

```
// Grauer Kegelstumpf im Nullpunkt
color("grey",.4)
translate([0, 10, 0]) rotate([0, 90, 0]) cylinder(h=10, r1=5, r2=2);

// Gespiegelter Kegelstumpf
mirror([10, 0, 0])
translate([0, 10, 0]) rotate([0, 90, 0]) cylinder(h=10, r1=5, r2=2);
```

Ein gespiegelter Kegelstumpf entlang der YZ-Fläche

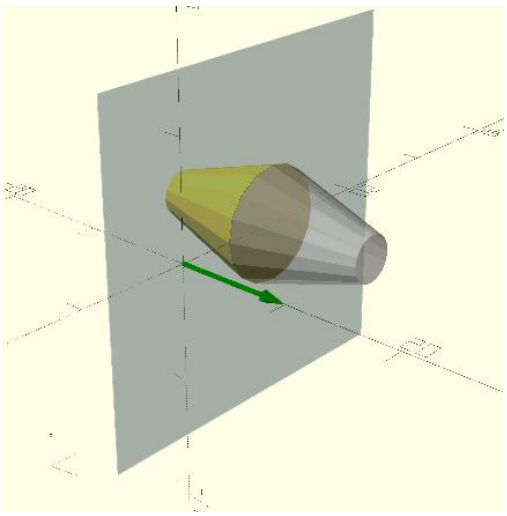
Mit einem grauen Nullpunkts-Kegelstumpf



Die Werte, die an Mirror übergeben werden, enthalten die Koordinaten für X, Y und Z. Diese enthalten die Angaben, die jeweils die Richtung der Spiegelung angeben.
Im Beispiel wird Z mit 10 bezeichnet, beide anderen enthalten 0. Somit wird die Reflektionsebene im rechten Winkel zur X-Achse dargestellt.

Zu beachten ist, dass Mirror **keine Kopie** erstellt. Wie z.B. beim Drehen ändert es nur die Position vom Objekt.

Um dies zu verdeutlichen, zeigt die Abbildung den „Spiegel“ als halbtransparente Ebene.



005-02

Der „Spiegel“ wird senkrecht zum Z-Wert, grau dargestellt, gezeichnet von (0, 0, 0) bis (10, 0, 0). Wichtig ist, dass der Wert 10 nicht als x-Achse verwendet wird, den Spiegel zu erstellen!

Jeder Wert auf der x-Achse ungleich Null würde die Spiegelung verursachen, da es nur eines anderen Wertes bedarf, damit der Spiegel senkrecht zur Achse steht. Die Spiegelebene enthält immer der Nullpunkt (0, 0, 0).

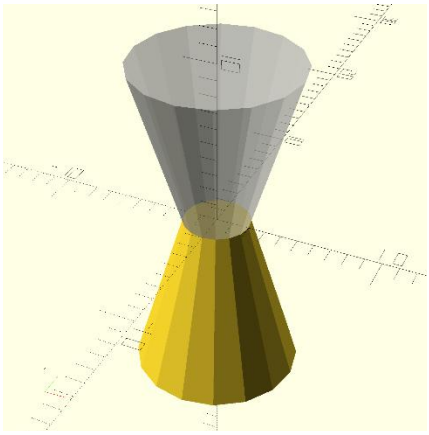
Der Wert beschreibt also wie der Spiegel gedreht wird.

Die nächste Anweisung spiegelt einen Zylinder über der XY-Ebene wider

```
// Grauer Kegelstumpf im Nullpunkt
color("grey",0.5) cylinder(h=10, r1=2, r2=5);

// Gespiegelter Kegelstumpf
mirror([0, 0, 10]) cylinder(h=10, r1=2, r2=5);
```

Ein gespiegelter Kegelstumpf zur Fläche von XY

Mit den Werten [0, 0, 10]



Die Spiegelfunktion ist zum schnellen Erstellen komplexer Formen mit Symmetrie besonders nützlich. Wenn „mirror“ in solchen Fällen zum Einsatz kommt, kann viel Zeit gespart werden, denn es wird nur die Hälfte gezeichnet und zusätzlich die andere Hälfte abschließend kopiert und gespiegelt!

## resize ( [ x, y, z ) ]

**Größenänderung/Skalierung der Form**

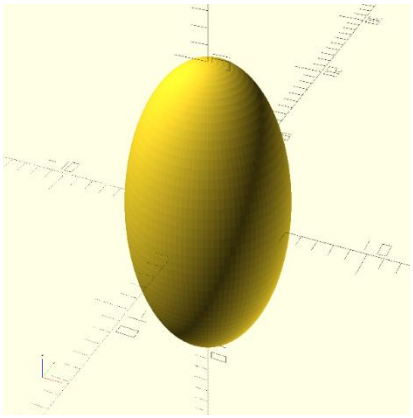
Mit der Skalierung können individuelle Formen mit bestimmten Abmessungen gedehnt oder verkleinert werden

Es kann jede Größe entlang jeder Achse der Form separat angegeben werden. Wird z.B. eine Kugel nur über eine Achse gestreckt, so kann diese in einen Ellipsoid (längliche Kugel) verwandelt werden.

Der folgende Code-Schnipsel verwendet „resize“, um aus einer Kugel ein Ellipsoid zu erstellen:

```
resize([10, 10, 20]) sphere(1, $fn=100);
```

Eine Kugel wurde in ein Ellipsoid verwandelt



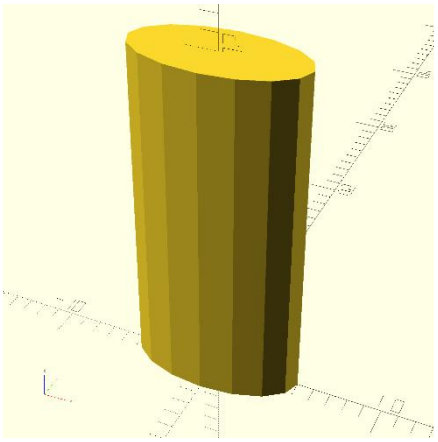
Nach der Eingabe Resize erfolgen die drei Größenangaben von X, Y und Z, komplett in ([ … ]) eingehüllt. Danach kommt erst die ursprüngliche Form.

Das neue Ellipsoid erstreckt sich:
5 Einheiten nach beiden Seiten des Nullpunkts entlang der X-Achse,
5 Einheiten nach beiden Seiten des Nullpunkts entlang der Y-Achse und
10 Einheiten nach beiden Seiten des Nullpunkts entlang der Z-Achse.

Die Größenänderung kann auch einen einfachen Zylinder verändern:

```
resize([10, 5, 20]) cylinder(h=5, r1=5, r2=5);
```

Hier werden beide Radien in der X-Achse auf 10 belassen, jedoch in der Y-Achse auf 5 gestaucht. Dafür wiederum in der Z-Achse auf 20 vergrößert.



005-04

## hull ( ) { ….. };

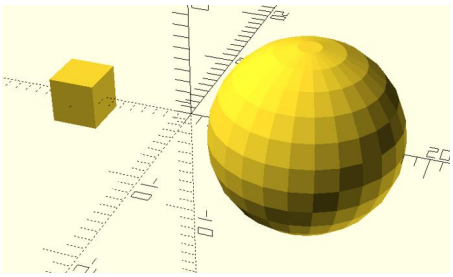
**Kombinieren von Formen mit einer Hülle**

Der Befehl „hull“ erzeugt eine konvexe Hülle (oder auch Haut) um zwei Formen, so als würde man einen Ballon um die Formen spannen.

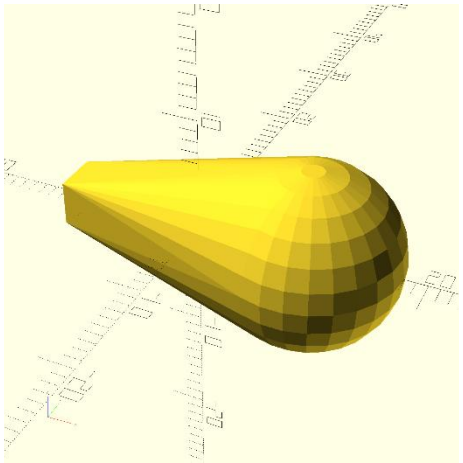
Beispiel: der folgende Code erzeugt einen Ballon, der sowohl eine Kugel als auch einen Würfel umgibt:

```
/*
    Funktion hull 1
*/

$fn=100;

hull() {
translate([10, 0, 0]) sphere(8);
translate([-10, 0, 0]) cube([4, 4, 4], center=true);
}
```

Ausgangsbild:

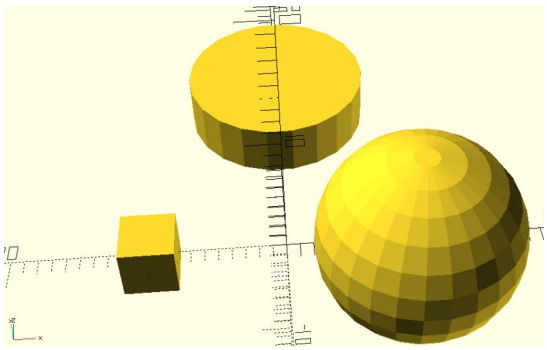


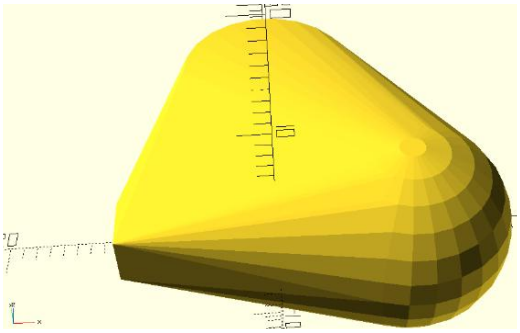
nachher:



Um den kleinen Würfel und die große Kugel spannt sich nun eine Hülle.

Die Hüllenoperation hat die gleiche Syntax wie die beschriebenen booleschen Operationen: es können zwei oder auch mehrere Formen kombiniert werden, und wie mit Vereinigungsoperation „union“ spielt die Reihenfolge der Formen keine Rolle.





```
/*
    Funktion hull 1
*/

$fn=100;

hull() {
translate([10, 0, 0]) sphere(8);
translate([-10, 0, 0]) cube([4, 4, 4], center=true);
}
```

Auch hier bei diesem Befehl werden die zu umschließende Objekte mit geschweiften Klammern eingefasst hull { }

005-06

## minkowski ( convexity );

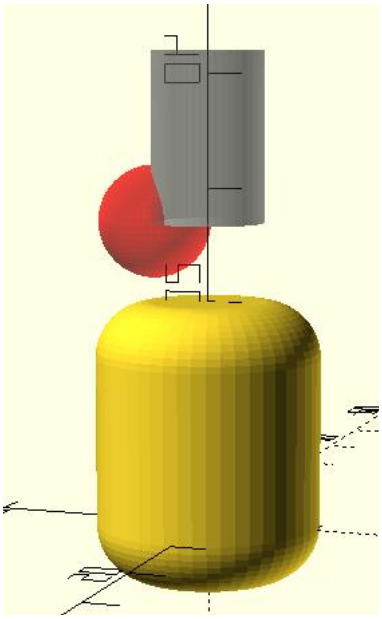
**Formen runden mit einer Kugel**

Der Befehl „minkowski“ erzeugt eine Minkowski-Summe, einer Sammlung von Formen. Dies hat zur Folge, dass die Kanten einer Form mit der Charakteristik einer zweiten Form vollständig umhüllt werden.

Zur Berechnung wird immer vom Mittelpunkt des umkreisenden Objektes ausgegangen.

Im folgenden Beispiel wird einem Zylinder um alle Kanten von einer Kugel umwickelt, damit abgerundete Kanten entstehen:

```
/*
    Funktion minkowski1
*/

$fn=50;
minkowski() {
cylinder(h=15, r=5);
sphere(5);
}

color("grey",0.5)
translate([0,0,27])
cylinder(h=15, r=5);

color("red",0.5)
translate([5,0,27])
sphere(5);
```

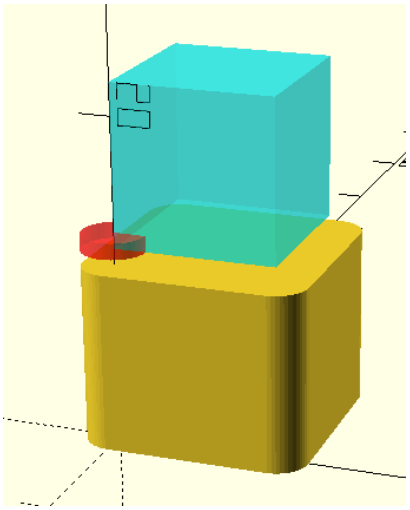


Der graue Zylinder hat die gleichen Abmessungen wie der eigentlich Gelbe. Um den gelben Zylinder müssen jedoch noch an allen Ecken und Kanten die Maße der umrundenden Kugel (Ø5) dazu gerechnet werden:

| Nullpunkts-Zylinder: | Minkowski-Zylinder |
|---|---|
| Höhe von 15 | 15 + je ½ Kugel oben + unten = 25 |
| Radius von 5 (Ø 10) | 10 + je ½ Kugel links + rechts = Ø 20 |

**minkowski mit umlaufenden Zylinder**

```
/*
    Funktion minkowski 2
*/

$fn=50;
minkowski()
{
  cube([10,10,10]);
  cylinder(r=2,h=1);
}

translate([0,0,17])
  color("Cyan",0.5)
    cube([10,10,10]);

translate([0,0,17])
  color("Red",0.5)
    cylinder(r=2,h=1);
```



Hier verläuft es etwas anders: Der kleine rote Zylinder umkreist den hellblauen Würfel an allen sechs Flächen. Dies ergibt den darunter liegenden gelben Würfel, dessen 4 Kanten gerundet sind.

Auch hier muss der Radius des Zylinders zweimal zur Kantenlänge des Würfels gerechnet werden:  10 + 2x Radius des Zylinders = 10 + 4 = 14.
Die äußeren Maße des Würfels sind nun 10 x 14.

Die Höhe setzt sich nun aus der eigentlichen hellblauen Würfelhöhe und zweimal der Hälfte der Höhe des Zylinders zusammen: 10 + 2x ½ Höhe = 10 + 1 = 11.

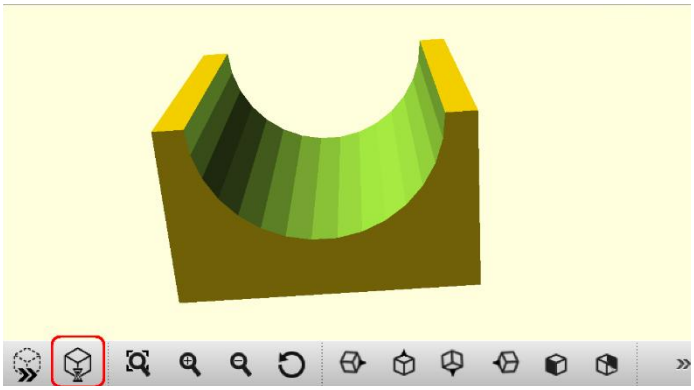
Es wird also immer entscheidend sein, welches Objekt mit welchen Ausmaßen in „minkowski“-Anweisungen eingesetzt werden.

005-08

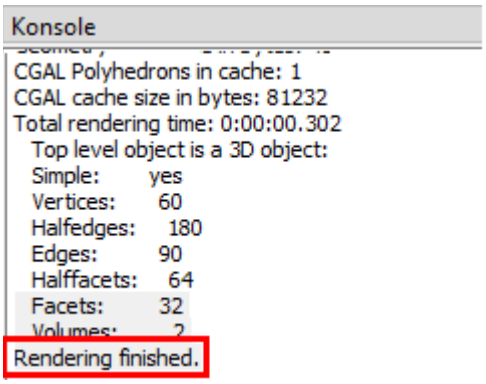
## Rendern / F 6 : Vorbereitung für den 3D-Druck

Ist die Gestaltung des Objektes abgeschlossen, kann es an eine weitere Anwendung (wie z.B. Cura) übergeben werden. Dazu muss es zuvor gerendert werden: dies ist eine genauere Berechnung aller Formen, Flächen  und Rundungen des Objektes als in der Vorschau benötigt wird.

Dazu wird auf die Render-Schaltfläche (im Bild rot markiert) geklickt:

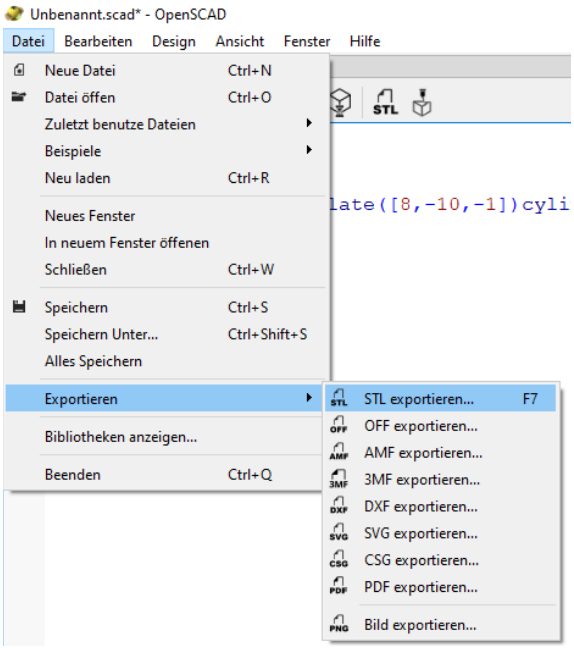


Besonders komplexe Konstruktionen erfordern selbstverständlich mehr Oberflächen und können daher längere Renderzeiten haben.
Einfach etwas mehr Zeit geben und warten bis in der Konsole „Rendering finished“ erscheint!




Nach dem Rendern kann das Objekt in eine STL-Datei abgespeichert werden.

Hierzu auf Datei – Exportieren und dann auf STL-exportieren (F7) klicken und einen treffendenden Namen eingeben.




**F 5 : Vorschau**

**F 6 : Rendern**

**F 7 : Exportieren als STL**