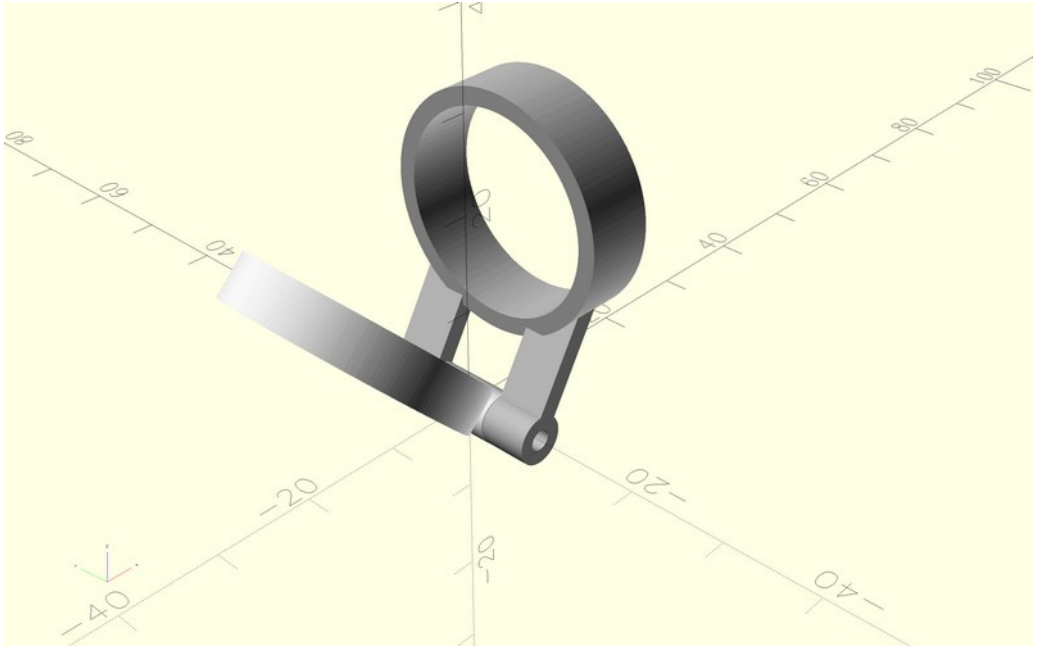
3D-Druck mit OpenSCAD



3D-Druck mit OpenSCAD

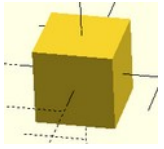
- Was ist openSCAD?
  - OpenSCAD ist eine Software zur Erstellung von 3D-CAD-Volumenmodellen. Es handelt sich um freie Software und steht unter der General Public License version 2.
  - Im Gegensatz zu den meisten freien Programmen zur Erstellung von 3D-Modellen (wie beispielsweise der bekannten Anwendung Blender) konzentriert sich OpenSCAD eher auf die CAD-Aspekte als auf die künstlerischen Aspekte der 3D-Modellierung.
  - OpenSCAD ist kein interaktiver Modellierer. Stattdessen ähnelt es eher einem 3D-Compiler, der eine Skriptdatei liest, die das Objekt beschreibt, und das 3D-Modell aus dieser Skriptdatei rendert.

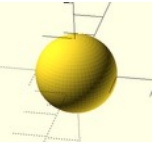
3D-Druck mit OpenSCAD

- Wie arbeitet openSCAD?
  - Als Designer bekommt man die vollständige Kontrolle über den Modellierungsprozess und ermöglicht dadurch, jeden Schritt im Modellierungsprozess einfach zu ändern oder Entwürfe zu erstellen, die durch konfigurierbare Parameter definiert sind.
  - OpenSCAD bietet zwei Hauptmodellierungstechniken:
    - Zum einen die konstruktive Volumengeometrie (auch bekannt als CSG - Constructive Solid Geometry).
    - Und zum anderen die Extrusion von 2D-Konturen.

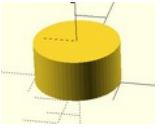
3D-Druck mit OpenSCAD

- Wie funktionieren die beiden Hauptmodellierungstechniken?





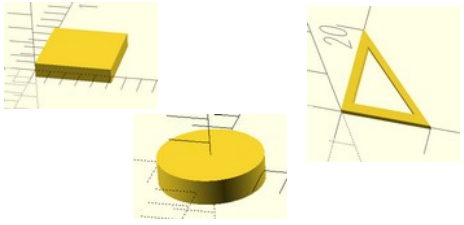
  - Die konstruktive Volumengeometrie:
    - ⇒ 3D Objekte können direkt erstellt werden wie:
      - Cube (Würfel) oder Rectangular prism (Rechteckiges Prisma)
      - Sphere (Kugel)
      - Cylinder (Zylinder)
      - Polyhedron (Polyeder / Vieleck)



    - ⇒ Die Dimensionen werden als Parameter dem Objekt beim erstellen zugewiesen

3D-Druck mit OpenSCAD

- Wie funktionieren die beiden Hauptmodellierungstechniken?



  - Extrusion von 2D-Konturen:
    - ⇒ 3D Objekte werden durch extrusion von 2D Konturen erzeugt
      - Square (Quadrat) wird transformiert in einen Würfel (Cube)
      - Circle (Kreis) wird transformiert in einen Zylinder (Cylinder)
      - Polygon (Vieleck) kann zu einem komplexen 3D Objekt transformiert werden
    - ⇒ In openSCAD sehen 2D Konturen wie 3D Objekte aus. Das täuscht, denn ihnen fehlt nach wie vor die 3. Dimension. Diese erhalten sie erst, nachdem die 2D Kontur durch Extrusion in ein 3D Objekt umgewandelt wird.

3D-Druck mit OpenSCAD

- OpenSCAD ist für verschiedene Betriebssysteme verfügbar
  - MacOS
  - Microsoft Windows
  - Linux
    - Debian/Ubuntu
    - Fedora
    - OpenSUSE
    - Arch Linux
    - und weiter Distributionen
  - BSD
    - NetBSD
    - FreeBSD
    - OpenBSD

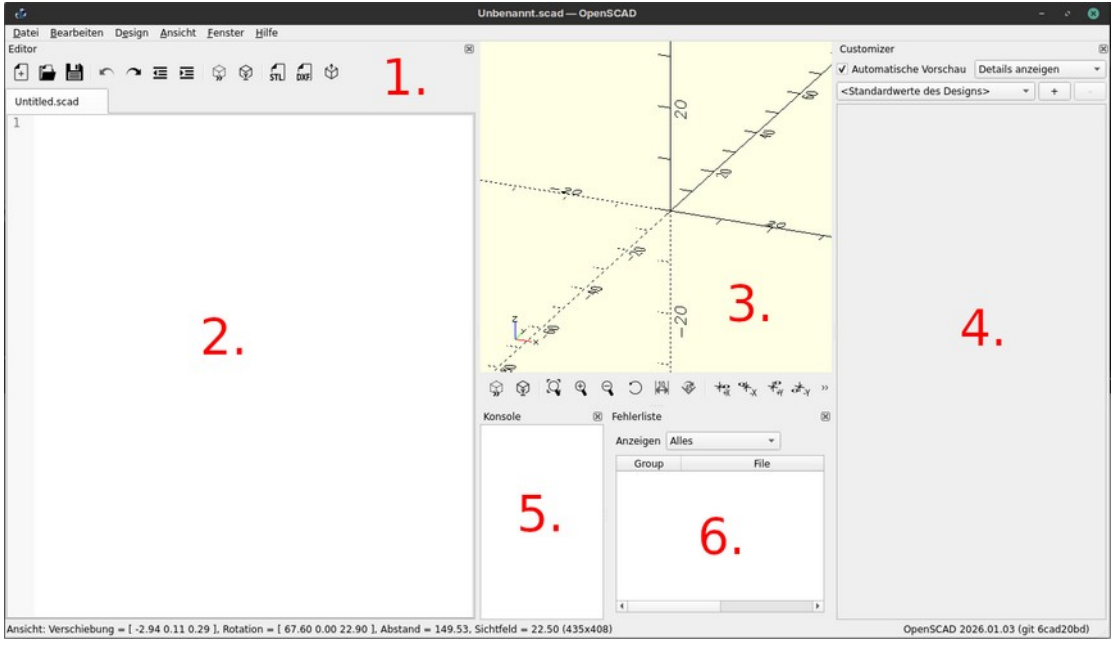
3D-Druck mit OpenSCAD

- OpenSCAD gibt es auch in Container
  - Docker
  - Flathub
  - Snap
  - AppImage
- Und natürlich auch als Source Code
- OpenSCAD Version
  - Standardmäßig ist die Version 2021.01 in den Repositories enthalten
  - Aktuell ist die Version 2026.05.31 verfügbar (Stand Juni 2026)

3D-Druck mit OpenSCAD

- Programmaufbau

1. Symbolleiste
2. Editor
3. Vorschau
4. Parametrierung
5. Konsolenausgabe
6. Fehlerliste/Warnung



3D-Druck mit OpenSCAD

- Syntax
  - Im Editorfenster (2.) kann frei programmiert werden.
  - OpenSCAD bietet Syntax-Highligting an.
    - ⇒ Bei Eingabe von z.B. pol schlägt das Programm die Typen vor
      - Polygon
      - Polyhedron
    - ⇒ Sobald nach dem gewünschten Objekt eine öffnende Klammer eingegeben wurde, erscheinen mögliche Parameterlisten und deren Eingabemöglichkeiten.

3D-Druck mit OpenSCAD

- Syntax
  - Kommentare werden mit // eingeleitet.
    - // Dies ist ein Kommentar
    - Cube(5); // Dies ist ein Kommentar
  - Kommentarblöcke werden in /* … */ eingefasst.
    - /* **************************
      *** Kommentarblock ***
      ************************** */
  - Die Größer/Kleiner Zeichen <> werden für das inkludieren von Dateien genutzt.
    - use </home/user/OpenSCAD/module.scad>
    - Include </home/user/.local/share/OpenSCAD/libraries/*/*.scad>
    - Include </usr/share/openscad/libraries/*/*.scad>

3D-Druck mit OpenSCAD

- Syntax
  - Runde Klammern () beschreiben eine Parameterliste.
    - (radius=r_kreis, center=true)
  - Eckige Klammern [] beschreiben eine Liste von Werten.
    - Koordinaten[x,y,z] oder Werteliste [a,b][c,d][e,f]
  - Geschweifte Klammern {} beschreiben Operatoren. Operatoren, auch Transformationen genannt, ändern die Position, Farbe und andere Eigenschaften von Objekten.
    - { square(size); circle(radius); }

3D-Druck mit OpenSCAD

- Syntax
  - Objekte werden mit einem Semikolon ; beendet.
    - circle(radius);
  - Das Komma , trennt Parameter oder Wertelisten voneinander.
    - cube(width, depth, high);
  - Der Punkt . ist für die Dezimalschreibweise.
    - circle(radius = 5.023);

3D-Druck mit OpenSCAD

- Syntax
  - Befehle können hintereinander in einer Zeile stehen.
    - ⇒ rotate([x,y,z]) circle(radius);
  - Leerzeichen werden im Programmcode ignoriert, solange die Objektnamen nicht getrennt werden.
  - Einrückungen dienen zu besseren Übersicht des Programmcodes.
  - Zahlen sind der wichtigste Wertetyp in OpenSCAD und werden in der bekannten Dezimalschreibweise angegeben, z. B. -1, 42, 0.5, 2.99792458e+8.

3D-Druck mit OpenSCAD

- Variablen
  - Ein Wert in OpenSCAD ist entweder
    - eine Zahl (wie 42)
    - ein Boolescher Wert (wie true)
    - eine Zeichenkette (wie „foo“)
    - ein Bereich (wie [0: 1: 10])
    - ein Vektor (wie [1,2,3])
    - oder der Wert „undef“ (undef).
  - Werte können in Variablen gespeichert, als Funktionsargumente übergeben und als Funktionsergebnisse zurückgegeben werden.

3D-Druck mit OpenSCAD

- Syntax
  - Variablen können definiert werden und sollten immer vor der ersten Verwendung angelegt werden.
  - Variablen erleichtern die Übersicht und die spätere Anpassbarkeit.
  - Variablen setzen im openSCAD keinen speziellen Datentypen voraus.
  - Variablen können Rechenoperationen wie z.B. + - * / enthalten.
    - corner = 2.5;
    - height = 31-2*corner;

3D-Druck mit OpenSCAD

- Syntax
  - Es können Funktionen mittels dem Befehl modules erstellt werden.
    - module rounded_square(height=10, width=20, radius_corner=1)
      {
      … Programmcode
      }
  - Aufgerufen werden die Funktionen wie alle Objekte mit Namen und Parameterliste sowie dem Semikolon.
    - rounded_square(height=20, width=20, radius_corner=2);
  - Nicht alle Parameter müssen übergeben werden, solange dem ausgelassenen Parameter ein Defaultwert zugewiesen wurde.
    - rounded_square(height=20, width=20);

3D-Druck mit OpenSCAD

- Syntax
  - Für wiederholende Objekte im Modell können Schleifen verwendet werden.
    - for (iter = [start:inkrement:ende])
      - iter → enthält den aktuellen Wert von einer Schleifenausführung.
      - start → definiert den Startwert
      - inkrement → definiert den Wert, der bei jedem Schleifendurchlauf jeweils erhöht wird
      - ende →definiert den Endwert

3D-Druck mit OpenSCAD

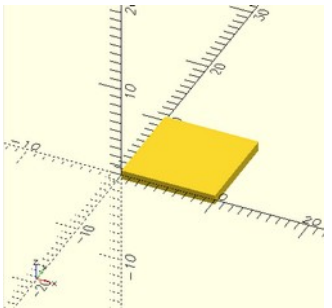
- 2D-Primitive
  - Alle 2D-Primitive können mit 3D-Transformationen transformiert werden. Sie werden üblicherweise als Teil einer 3D-Extrusion verwendet. Obwohl sie unendlich dünn sind, werden sie mit einer Dicke von 1 Einheit gerendert.
  - 2D-Objekte können auf verschiedenster Art transformiert werden.
    - ⇒ Ein 2D-Objekt kann mittels „linear_extrude()“ in die länge gezogen werden.
    - ⇒ Ein 2D-Objekt kann auch durch „rotate_extrude()“ in ein Rotationsobjekt verändert werden.

3D-Druck mit OpenSCAD

- Quadrat
  - Angaben
    - ⇒ Kantenlänge für die X-Achse
    - ⇒ Kantenlänge für die Y-Achse

```
square(size = [x, y], center = true/false);
square(size =  x    , center = true/false);
```

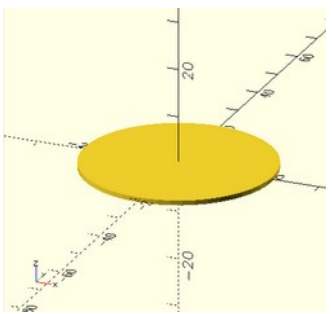
```
square(size = [1, 1], center = false);

Equivalente Syntax für dieses Beispiel
 square(size = 10);
 square(10);
 square([10,10]);
 square(10,false);
 square([10,10],false);
 square([10,10],center=false);
 square(size = [10, 10], center = false);
 square(center = false,size = [10, 10] );
```



- Kreis
  - Angaben
    - ⇒ Radius | Durchmesser

```
circle(r=radius | d=diameter);
```

```
// erstellt einen Kreis mit radius
   von 20 in hoher Auflösung.
scale([1/100, 1/100, 1/100])circle(2000);

// Anderer Ansatz.
circle(20, $fn=50);

Equivalente Syntax für dieses Beispiel
 circle(20);
 circle(r=20);
 circle(d=40);
 circle(d=4+18*2);
```



3D-Druck mit OpenSCAD

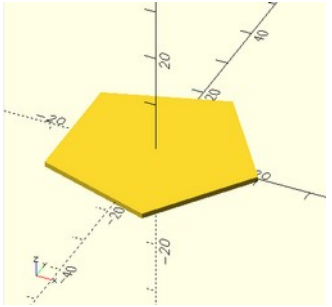
- Regelmäßige Vielecke
  - Angaben
    - Radius
    - Anzahl Kanten / Ecken

```
circle(r=radius, $fn=Anzahl Kanten)
```

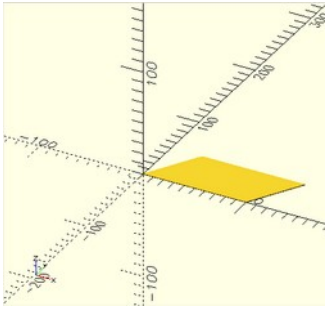
- Vielecke
  - Angaben
    - Punkte
    - Pfade
    - Konvexität

```
polygon(points=[ [x,y],... ] \
    ,paths=[[p1,p2,p3..],...],convexity=N);
```

```
circle(r=20, $fn=5);

Equivalente Syntax für dieses Beispiel
 translate([-42,  0]) circle(20,$fn=3);
 translate([  0,  0]) circle(20,$fn=4);
 translate([ 42,  0]) circle(20,$fn=5);
 translate([-42,-42]) circle(20,$fn=6);
 translate([  0,-42]) circle(20,$fn=8);

 color("black"){
     translate([-42,  0,1])text("3",7);
     translate([  0,  0,1])text("4",7);
     translate([ 42,  0,1])text("5",7);
     translate([-42,-42,1])text("6",7);
     translate([  0,-42,1])text("8",7);
 }
```



```
polygon([[0,0],[100,0],[130,50],[30,50]]);

Equivalente Syntax für dieses Beispiel
 Polygon([[0,0],[100,0],[130,50], \
         [30,50]], paths=[[0,1,2,3]]);
 Polygon([[0,0],[100,0],[130,50], \
         [30,50]],[[3,2,1,0]]);
 Polygon([[0,0],[100,0],[130,50], \
         [30,50]],[[1,0,3,2]]);

 a=[[0,0],[100,0],[130,50],[30,50]];
 b=[[3,0,1,2]];
 polygon(a);
 polygon(a,b);
 polygon(a,[[2,3,0,1,2]]);
```

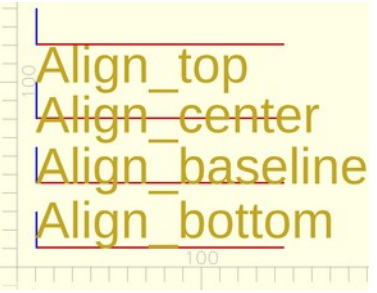


3D-Druck mit OpenSCAD

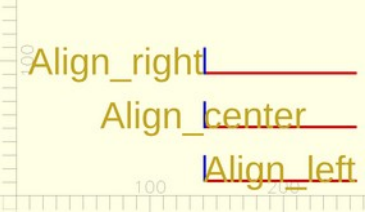
- Texte
  - Angaben
    - ⇒ Text
    - ⇒ Schriftgröße
    - ⇒ Schriftart
    - ⇒ Hor. Ausrichtung
    - ⇒ Ver. Ausrichtung
    - ⇒ Schreibrichtung
    - ⇒ Zeichenabstand
    - ⇒ Sprache

```
text(text="",size=1,font="",halign="",
     valign="",direction="",spacing=1,
     language="",script="");
```

```
text = "Align";
font = "Liberation Sans:style=Regular";
 valign = [
  [  0, "top"],
  [ 40, "center"],
  [ 75, "baseline"],
  [110, "bottom"]];
 for (a = valign) {
  translate([10, 120 - a[0], 0]) {
   color("red") cube([135, 1, 0.1]);
   color("blue") cube([1, 20, 0.1]);
   linear_extrude(height = 0.5) {
     text(text = str(text,"_",a[1]), \
          font=font,size=20,valign=a[1]);
} } }
```




```
text = "Align";
font = "Liberation Sans:style=Regular";
 halign = [
  [10, "left"],
  [50, "center"],
  [90, "right"]
];
 for (a = halign) {
  translate([140, a[0], 0]) {
    color("red") cube([115, 2,0.1]);
    color("blue") cube([2, 20,0.1]);
    linear_extrude(height = 0.5) {
      text(text = str(text,"_",a[1]), \
      font=font,size=20,halign=a[1]);
} } }
```



3D-Druck mit OpenSCAD

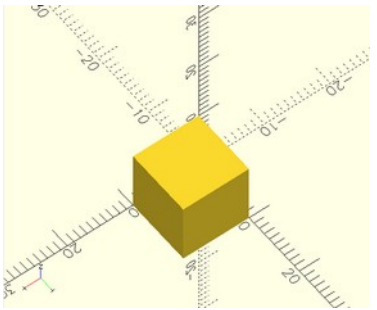
- 3D-Objekte
  - Alle 3D-Objekte sind, im Gegensatz zu den zuvor beschriebenen 2D- Primitives, die mit Zusatzfunktionen erst in ein 3D- Objekt umgewandelt werden, mit einem Befehl erstellbar.

3D-Druck mit OpenSCAD

- Würfel
  - Angaben
    - ⇒ Kantenlänge für die X-Achse
    - ⇒ Kantenlänge für die Y-Achse
    - ⇒ Kantenlänge für die Z-Achse

```
cube(size);
cube(width,depth,height, center=true/false);
```

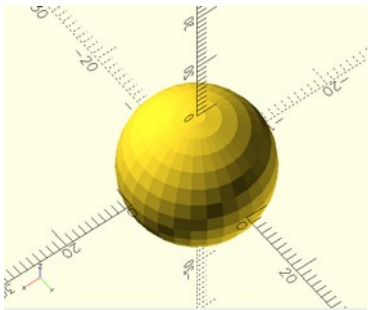
```
cube(size = [10,10, 10], center = false);

Equivalente Syntax für dieses Beispiel
 cube(size = 10);
 cube(10);
 cube([10,10,10]);
 .
 cube(10,false);
 cube([10,10,10],false);
 cube([10,10,10],center=false);
 cube(size = [10,10,10], center = false);
 cube(center = false,size = [10,10,10] );
```



- Kugel
  - Angaben
    - ⇒ Radius | Durchmesser

```
sphere(r=radius | d=diameter);
```

```
// erstellt eine Kugel mit radius
   von 10 in hoher Auflösung.
sphere($fn = 0, $fa = 12, $fs = 2, r = 10);

// Anderer Ansatz.
$fn = 0; $fa = 12; $fs = 2;
sphere(r = 10);

Equivalente Syntax für dieses Beispiel
 sphere(20);
 sphere(r=20);
 sphere(d=40);
 sphere(d=4+18*2);
```

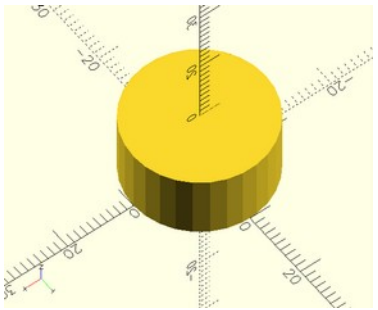


3D-Druck mit OpenSCAD

- Zylinder
  - Angaben
    - Höhe
    - Unterer Radius
    - Oberer Radius

```
cylinder(h = height, r1 = BottomRadius,
      r2 = TopRadius, center = true/false);
```

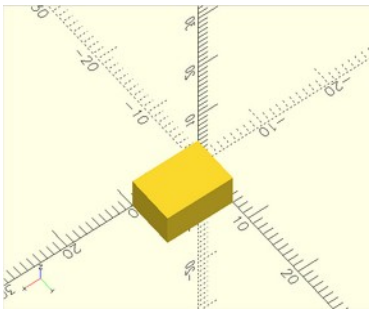
```
cylinder($fn = 0, $fa = 12, $fs = 2,
h = 10, r1 = 10, r2 = 10, center = false);

Equivalente Syntax für dieses Beispiel
 cylinder(h=15, r1=9.5, r2=19.5, false);
 cylinder(  15,    9.5,    19.5, false);
 cylinder(  15,    9.5,    19.5);
 cylinder(  15,    9.5, d2=39  );
 cylinder(  15, d1=19,  d2=39  );
 cylinder(  15, d1=19,  r2=19.5);
```



- Polyeder
  - Angaben
    - Vektor aus Eckpunkte
    - Vektor aus Flächen
    - Konvexität

```
polyhedron( points = [ [X0, Y0, Z0], ... ],
faces = [ [P0, P1, P2, P3, ...], ...],
convexity = N);
```

```
CubePoints = [
  [  0,  0,  0 ],  //0
  [ 10,  0,  0 ],  //1
  [ 10,  7,  0 ],  //2
  [  0,  7,  0 ],  //3
  [  0,  0,  5 ],  //4
  [ 10,  0,  5 ],  //5
  [ 10,  7,  5 ],  //6
  [  0,  7,  5 ]]; //7
CubeFaces = [
  [0,1,2,3],  // bottom
  [4,5,1,0],  // front
  [7,6,5,4],  // top
  [5,6,2,1],  // right
  [6,7,3,2],  // back
  [7,4,0,3]]; // left
polyhedron( CubePoints, CubeFaces );
```



3D-Druck mit OpenSCAD

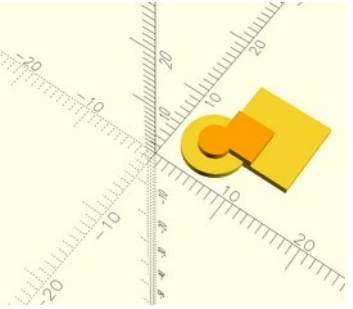
- Sonderfunktionen
  - Mit Sonderfunktionen können erstellte Objekte verändert werden.
    - translate{[]} → Damit Formen und Objekte im 3-dimensionalem Raum ihre korrekte Position finden, können diese von ihrer Ursprungskoordinate X=0, Y=0, Z=0 auf eine neue Koordinate übertragen / umgewandelt werden.
    - difference() → Mit dieser Funktion wird vom ersten Objekt ein zweites Objekt subtrahiert. Somit lassen sich verschiedene Objekte „beschneiden“ und so zum Beispiel ein Rohr erstellt werden.
    - linear_extrude() → Um 2D-Formen in ein 3D Objekte umzuwandeln, werden diese extrudiert und somit um einer 3. Dimension erweitert.
    - rotate_extrude() → Bei der Rotationsextrusion wird eine 2D-Form um die Z-Achse gedreht, um einen Körper mit Rotationssymmetrie zu erzeugen.

3D-Druck mit OpenSCAD

- Übertragung / Umwandlung
  - Angaben
    - ⇒ Koordinatenwerte x,y,z

```
translate{[x,y,z]} {…}
```

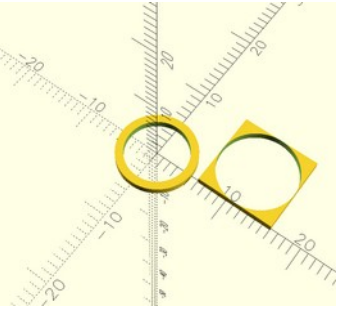
```
color("orange"){
translate([5,5,1]) square(5);
translate([5,5,1]) circle(2);
}

translate([5,5,0]) {
    square(10);
    circle(4);
}
```



- Differenzierung
  - Angaben
    - ⇒ Keine

```
difference() {…}
```

```
difference() {
    circle(5);
    circle(4);
}

difference() {
translate([6,0,0])
       square(10);
translate([11,5,0])
       circle(5);
}
```

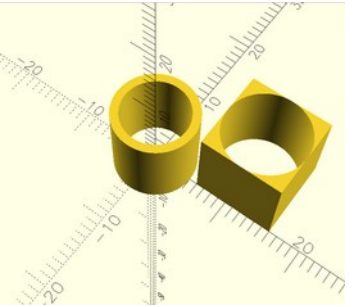


3D-Druck mit OpenSCAD

- Lineare Extrudierung
  - Angaben
    - Höhe
    - Zentrierung
    - Konvexität
    - Drehung
    - Skalierung

```
linear_extrude(height,center, \
               convexity,twist,scale) {…}
```

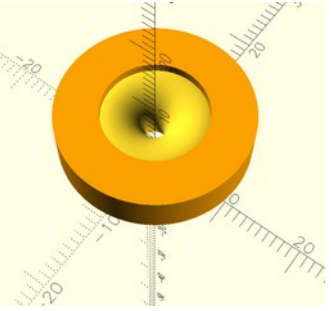
```
linear_extrude(height=10, center=false, \
convexity=1, twist=0, scale=1.0) {
    difference() {
        circle(5);
        circle(4);
}   }

linear_extrude(height=10, center=false, \
convexity=1, twist=0, scale=1.0) {
    difference() {
    translate([6,0,0])square(10);
    translate([11,5,0])circle(5);
}   }
```



- Rotierende Extrudierung
  - Angaben
    - ⇒ Winkel
    - ⇒ Konvexität

```
rotate_extrude(angle=360, convexity=1) {…}
```

```
color("orange"){
rotate_extrude(angle=360, \
               convexity=1) {
translate([6,5,1])
    square(5);
}   }

rotate_extrude(angle=360, \
               convexity=1) {
translate([5,5,0]) {
    circle(4);
}    }
```



3D-Druck mit OpenSCAD

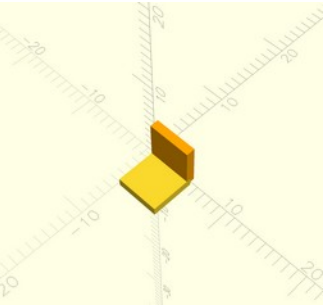
- Sonderfunktionen
  - Mit Sonderfunktionen können erstellte Objekte verändert werden.
    - rotate() → Damit Formen und Objekte im 3-dimensionalem Raum ihre korrekte Position finden, können diese auch von ihrer Ursprungsachse X=0°, Y=0°, Z=0° auf eine neue Position gedreht werden.
    - color() → Einzelne oder gruppierte Formen und Objekte können farblich im Modell hervorgehoben. Dies unterstützt den Anwender bei der visualisierten Erstellung von Modellen.
    - projection() → Es kann eine 3D-Form in ein 2D Objekt umgewandelt werden. Dadurch werden die Grundflächen des 3D- Modells sichtbar.

3D-Druck mit OpenSCAD

- Drehung
  - Angaben
    - ⇒ Drehungsachse x,y,z

```
rotate([x,y,z]) {…}
```

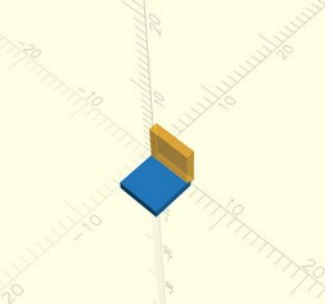
```
color("orange") {
rotate([90,0,0]) square(5);
}

rotate([0,0,-90]) {
    square(5);
}
```



- Farbe
  - Angaben
    - ⇒ Farbton
    - ⇒ Alphawert (Transparenz)

```
color(c = [r,g,b,a]) {…}
color(„colorname“, alpha) {…}
```

```
color("orange", alpha=0.5) {
rotate([90,0,0]) square(5);
}

color("#2277BB") {
rotate([0,0,-90]) {
    square(5);
}   }
```



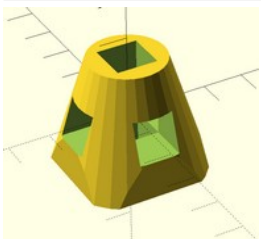
3D-Druck mit OpenSCAD

- 3D zu 2D Projektion
  - Angaben
    - ⇒ Schnitt
- Wenn cut = true ist, werden nur Punkte mit z=0 berücksichtigt (wodurch das Objekt effektiv abgeschnitten wird)
- Wenn cut = false (der Standard-einstellung) werden auch Punkte oberhalb und unterhalb der Ebene berücksichtigt (wodurch eine korrekte Projektion entsteht).

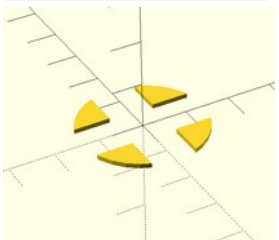
```
projection(cut = true/false) {...}
```

```
module example002()
{
  intersection() {
    difference() {
      union() {
        cube([30, 30, 30], center=true);
        translate([0, 0, -25])
          cube([15, 15, 50], center=true);
      }
      union() {
        cube([50, 10, 10], center = true);
        cube([10, 50, 10], center = true);
        cube([10, 10, 50], center = true);
      }
    }
    translate([0, 0, 5])
      cylinder(h=50,r1=20,r2=5,center=true);
  }
} // Written by Clifford Wolf <clifford@clifford.at> Marius Kintel <marius@kintel.net>
```

```
exampel002();
```



```
projection(cut = true)
exampel002();
```



```
projection(cut = false)
exampel002();
```

3D-Druck mit OpenSCAD

- Sonderfunktionen
  - Spezielle Variablen beeinflussen die 3D-Formen und Objekte
    - $fa → (minimum angle) ist der Mindestwinkel für ein Liniensegment. Selbst ein riesiger Kreis hat nicht mehr als 360 geteilt durch diese Zahl Liniensegmente. Der Standardwert ist 12 (d. h. 30 Liniensegmente für einen vollständigen Kreis). Der kleinste zulässige Wert ist 0,01.
    - $fs → (minimum size) ist die Mindestlänge eines Liniensegments. Der Standardwert beträgt 2, sodass sehr kleine Kreise eine geringere Anzahl an Liniensegmenten aufweisen als durch die Angabe von $fa festgelegt. Der zulässige Mindestwert beträgt 0,01.
    - $fn → (number of segments) ist die Anzahl der Liniensegmente und hat normalerweise den Standardwert 0. Wenn diese Variable einen Wert größer als Null hat, werden die beiden anderen Variablen ignoriert, und es wird ein voller Kreis mit dieser Anzahl von Liniensegmenten gezeichnet.
- Es gibt noch eine große Anzahl weiterer Funktionen und Variablen.
- Schaue hierzu ins Cheatsheet von OpenSCAD

3D-Druck mit OpenSCAD

- Nachdem nun das Modell erstellt wurde, erfolgt der nächste Schritt für den 3D-Druck
  - Das Modell muss in ein anderes Format exportiert werden.
  - Üblicherweise bietet openSCAD die Formate .stl und .dxf an.
    - .stl → Stereolithografie-Format (Geometrie eines dreidimensionalen Datenmodells)
    - .dxf → Drawing Interchange Format (CAD Format)
  - OpenSCAD bietet aber noch mehr Export-Formate an.

3D-Druck mit OpenSCAD

- Nachdem erfolgreich ein Modell erstellt wurde, kann mit der Vorbereitung für den 3D-Druck begonnen werden.
- Hierfür wird eine gesonderte Software benötigt, die das erstellte Modell mittels Slicer in ein druckbares Modell umwandelt. Danach wird das Modell in eine vom 3D-Drucker verständliche Datei übersetzt.
- Auf dem Markt gibt es eine größere Anzahl an Software hierzu.
  - PrusaSlicer
  - OrcaSlicer
  - Und viele andere
- Als Slicer-Software, kurz Slicer, oder Slicing Software (von engl. „to slice“ für „in Scheiben schneiden“) wird eine Software bezeichnet, die in den meisten 3D-Druckprozessen zur Umwandlung eines 3D-Objektmodells in spezifische Anweisungen für den Drucker verwendet wird. Insbesondere die Konvertierung von einem Modell im STL-Format in Druckerbefehle im G-Code-Format bei der Fused Deposition Modeling und anderen ähnlichen Prozessen.
- Die Auswahl des Slicers hängt von verschiedenen Faktoren ab. Zum einen ist es wichtig, dass die Software den verwendeten 3D-Drucker unterstützt und zum anderen von den eigenen Anforderungen / Vorlieben an die Software.

3D-Druck mit OpenSCAD

- PrusaSlicer
  - Die Software basiert auf den von Alessandro Ranelluccis programmierten Slic3r
  - Wird von der Firma Prusa Research als OpenSource Projekt weiter entwickelt
  - Prusa Research stellt eigene 3D- Drucker her
- SuperSlicer
  - Fork von PrusaSlicer
  - Open Source Projekt
- OrcaSlicer
  - Fork von BambuStudio
  - Open Source Projekt
  - Unterstützt die meisten 3D Drucker auf dem Markt
- BambuStudio
  - Fork von PrusaSlicer
  - Kommt von der Firma Bambu Lab
  - Bambu Lab stellt eigene 3D- Drucker her

3D-Druck mit OpenSCAD

- Tipps für den Anfang
  - Verschiedene Tutorials durcharbeiten (siehe zum Beispiel in openscad.org)
  - Mit kleinen Projekten anfangen
  - Dran bleiben

- Notwendige Informationen zu dem Thema findet ihr in den nachfolgenden Links.
- Dort findet ihr, dank der Autoren, viele nützliche Tipps, wie ihr erfolgreich zu eurem eigenen 3D-Druck Modell kommt.

3D-Druck mit OpenSCAD

- Links
  - https://en.wikipedia.org/wiki/OpenSCAD
  - https://openscad.org/documentation.html
  - https://openscad.org/cheatsheet/
  - https://en.wikibooks.org/wiki/OpenSCAD_User_Manual
  - https://openscad-meistern.de/
  - https://www.accessiblestem.org/openscad/libraries.html
  - https://gusmith.wordpress.com/
  - https://github.com/revarbat/BOSL
  - https://github.com/BelfrySCAD/BOSL2
  - https://www.printables.com

3D-Druck mit OpenSCAD

- Links
  - https://de.wikipedia.org/wiki/Slicer-Software
  - https://de.wikipedia.org/wiki/Slic3r
  - https://slic3r.org
  - https://de.wikipedia.org/wiki/Prusa_Research
  - https://www.prusa3d.com/de/p/prusaslicer
  - https://www.orcaslicer.com
  - https://superslicer.net
  - https://bambulab.com/de-de/download/studio