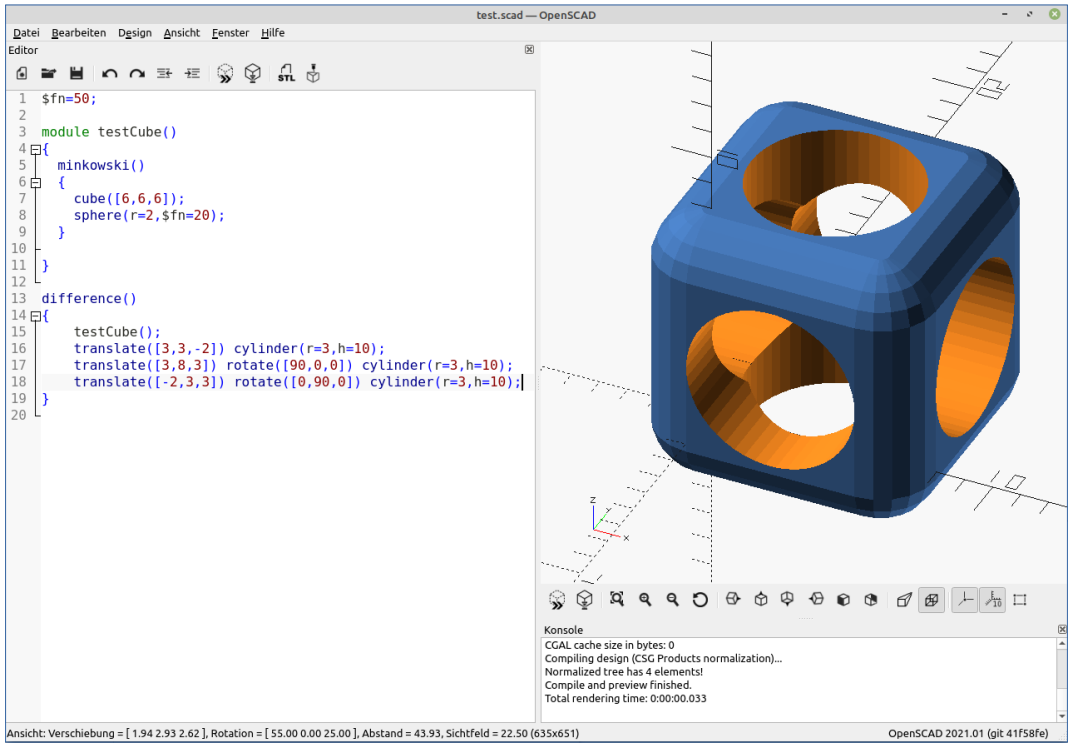
Skriptbasiertes CAD mit OpenSCAD

für den 3D Druck



Ersteller: andimoto (www.github.com/andimoto) - September 2023

Bevor es los geht...

- OpenSCAD installiert?
- Basiskurs angeschaut?
- (optional) Lieblingseditor gestartet?

Wiederholung

Aufbau der Syntax

- Objekte
  - 2D oder 3D Objekte
  - Objekte schließen immer mit ; ab
  - *Beispiele:*

Form	Beschreibung
`cube([20,20,20]);`	Kubus – alle Kanten 20mm
`sphere(50);`	Kugel mit Radius 50mm
`cylinder(r=2,h=50);`	Zylinder mit Radius 2mm & Höhe 50mm
`cylinder(r1=2,r2=4,h=50);`	Zylinder – unten 2mm, oben 4mm
`polygon(points=[[0,0],[0,1],[1,1]]);`	2D Polygon Fläche mit Koordinaten
`circle(d=30);`	2D Kreis mit Durchmesser 30mm

Wiederholung

Aufbau der Syntax

Datentyp	Beschreibung
`Nummern (numbers)`	Jede Art von Nummer – 1,2,3, PI, 1.02, etc.
`Bool‘sche Werte`	Typ mit 2 möglichen Werten – Wahr oder Falsch (true/false). Werte für Falsch: **false, 0, -0, „“, [], undef**. Alle anderen Werte sind Wahr (sie existieren)
`Strings`	Das ist ein String `s=„OpenSCAD!“;`
`Bereiche (ranges)`	Bereiche geben Start und Ende als auch die Schrittweite an. Mit Nummern. Bsp: `[start:ende]` oder `[start:inkrement:ende]`
`undef`	Nicht definiert. z.B. eine fehlende Variable
`----------`	`----------`
`Variablen`	Variablen dürfen aus folgenden Zeichen bestehen: **[a-zA-Z0-9_]**

Wiederholung

Aufbau der Syntax

- Transformationen (Operators)
  - Modifizieren Ort, Ausrichtung, Farbe, etc. eines oder mehrerer Objekte
  - Stehen immer vor einem Objekt (schließt NICHT mit ; ab)
  - Ist das „Adjektiv“ des Objekts
  - Beispiele:

Form und Transformation	Beschreibung
`translate([x,y,z]) cube([sx,sy,sz]);`	Kubus um x,y,z verschoben
`scale([2,1,1]) sphere(50);`	Kugel mit Radius 50mm um Faktor 2 in x Richtung skaliert
`rotate([0,45,0]) cylinder(r=2,h=50);`	Zylinder, gedreht um 45° um die y Achse

Wiederholung

Aufbau der Syntax

- Module
  - module definiert ein Objekt
  - Abgeschlossene, wiederkehrende Objekte
  - analog zur Funktion/Methode in anderen Programmiersprachen
  - Kann mehrere Parameter (mit Standard Wert) haben
  - Bei Aufruf ohne Parameter werden die „default“-Werte verwendet
  - Syntax:

```
module teil001(parameter1=“default Wert“,    p2=33)
{
    cube(parameter1);
    translate([parameter1]) cylinder(r=parameter1,h=p2);
}
```

Gut zu Wissen

- Die Variable **`$fn`**
  - Bestimmt die Anzahl der Polygone die für das Modell verwendet werden
    - Bsp: `$fn = 100;`
  - Je höher der Wert, desto detaillierter das Modell, desto länger benötigt das Rendern (oder auch die Vorschau)
  - Vorsichtig damit!! :)
- Debugging (Fehlersuche)
  - # vor ein Objekt markiert dieses rot
  - % vor ein Objekt deaktiviert dieses und markiert es grau
  - ! vor ein Objekt um nur dieses Objekt anzuzeigen

Transformationen

Transformation	Beschreibung
`hull(){...};`	Legt eine Hülle um die angegebenen Objekte
`minkowski(){p1,p2};`	Rechnet Objekt 2 zu Objekt 1 hinzu. Bsp. Kanten eines Würfels abrunden
`resize([x,y,z]) object();`	Vergrößert das Objekt auf die angegebenen Absolutwerte
`mirror([x,y,z]) object();`	Spiegelt das Objekt entlang der ausgewählten Achsen. Mit ‚1‘ wird die Auswahl gesetzt

Weitere Funktionen

Objekt als Transformation (2D → 3D)	Beschreibung
`linear_extrude(height, center, convexity, twist) 2d_object();`	Macht aus einem 2D Objekt ein 3D Objekt mit einer Höhe. 2D Objekte liegen auf xy Ebene und werden in z Richtung extrudiert.
`rotate_extrude(angle, convexity) 2d_object();`	Extrudiert das 2D Objekt um die z Achse in einem angegebenen Winkel (oder 360°). 2D Objekt wird dazu auf xz Ebene „aufgestellt“.

Anmerkung (nicht Teil dieses Workshops):

- Die Transformation „rotate_extrude()“ wird nicht weiter verwendet
- „convexity“ dient nur zur korrekten Darstellung und hat keinen Einfluss auf das Modell
- „center“ bestimmt die Lage auf der xy Ebene. Wert „false“ legt Bbjekt **auf** die xy Ebene. Wert „true“ legt xy Ebene mittig durch das Objekt

Bool'sche Operationen

Bool'sche Operation	Beschreibung
`union() { p1(); p2(); ...}`	**ODER** Operation. Gruppiert alle inneren Objekte zu einer Einheit.
`difference() { p1(); p2(); ...}`	**UND NICHT** Operation. Zieht vom 1. Objekt alle weiteren Objekte ab, welche die Form des 1. Objekts kreuzen.
`intersection() { p1(); p2(); ...}`	**UND** Operation. Die kreuzenden Volumen aller inneren Objekt bilden eine Einheit → neues Objekt.

Anmerkung: Alle weiteren Transformationen und Operationen werden auf die gesamte Einheit angewendet. Jede bool'sche Operation bildet, anders gesagt, ein Objekt (welches aber nicht aufgerufen werden kann).

Platzierung mit Listen und Schleifen

- Die Platzierung von mehreren Objekten kann kurz und elegant durch eine Liste erreicht werden
- In einer **for-Schleife** werden dann die einzelnen Objekte an ihre Position verschoben

Befehl	Beschreibung
`pos = [[x1,y1], [x2,y2], [x3,y3]];`	Liste hält Koordinaten für alle Objekte

`for( p = pos )`
`{`
`  translate([p[0],p[1],0)`
`  cylinder(...);`
`}`
	Der Variablen p wird in jeder Iteration ein Wert der Liste pos zugewiesen.

Übung: Schleife und Liste

- es soll eine Liste mit 4 xy Koordinaten angelegt werden
  - Koordinaten sollen aus bestehenden Variablen „berechnet“ werden
- es sollen 4 Schrauben (module screw()) entlang der oberen Fläche platziert werden
  - verwende dafür die Datei „makerspace-box-004-list-for.scad“

Bedingungen

- Bedingungen erlauben eine selektive Platzierung oder Erstellung von Objekten
- Mit einer if oder if-else Bedingung können verschiedene Eigenschaften von Objekten selektiert werden
  - Achtung: Zuweisung von Variablen gilt nur für den Scope. Danach nimmt die Variable den ursprünglichen Wert an

Befehl	Beschreibung

`if(placeScrews == true)`
`{`
`  screw();`
`}`
	Durch die Abfrage einer Variablen kann ein Objekt platziert werden oder nicht.

`i = 5;`
`if (select == true) {`
`  i = 10;`
`}`
	<- i hat den Wert 5
	<- neuer Scope beginnt mit {
	<- i hat den Wert 10
	<- Scope endet mit } - i ist nun 5

Bedingungen

- Da Zuweisungen nach dem Ende des Scopes verloren gehen, kann man keine Variable bedingt setzen und in einer nächsten Anweisung verwenden
  - Abhilfe schaffen temporäre Variablen oder Funktionen

Befehl	Beschreibung
`function func3(y=7)=(y==7) ? 5 : 2;`	Eine Abfrage kann in eine Funktion verschoben werden. Funktionen können auch direkt in Operationen usw. verwendet werden

Übung: Bedingungen

- Verschiedene Aufrufe von Modulen soll durch bool‘sche Variablen und Bedingungen stattfinden
  - Im Customizer von OpenSCAD kann ein Objekt oder Modul ein- oder ausgeschaltet werden
  - Das Modul „assembly“ und die „animations“-Module sollen per if-Abfrage auswählbar gemacht werden
- Durch die Abfrage einer bool‘schen Variable sollen verschiedene Durchmesser von Schrauben oder Einsatzmuttern eingefügt werden
- Verwende die Datei 002-makerspace-box-006-if.scad und die Datei 002-makerspace-box-006-func.scad für diese Aufgabe

Import von SVG oder STL

- Es können SVG Dateien (Vektorgrafik) importiert werden
  - Diese liegen dann als 2D Fläche vor und müssen extrudiert werden
- STL Dateien (gerenderte 3D Modelle) können importiert und bearbeitet werden
  - Importierte 3D Modelle können dann verändert werden

Befehl	Beschreibung
`import("grafik.svg", convexity=3);`	Importiert 2D Vektorgrafik. Erlaubt sind SVG oder DXF.
`import("datei.stl", convexity=3);`	Importiert STL oder 3MF. Mit „convexity“ können Anzeigefehler behoben werden.
surface(file = "smiley.png", center = true, invert = true);	Aus PNG Dateien kann OpenSCAD Graustufen machen und in ein 3D Model extrudieren

Das wars!!