27. März 2023

Funktionales 3D-Design mit OpenSCAD

Lukas Wachter, Universität des Saarlandes

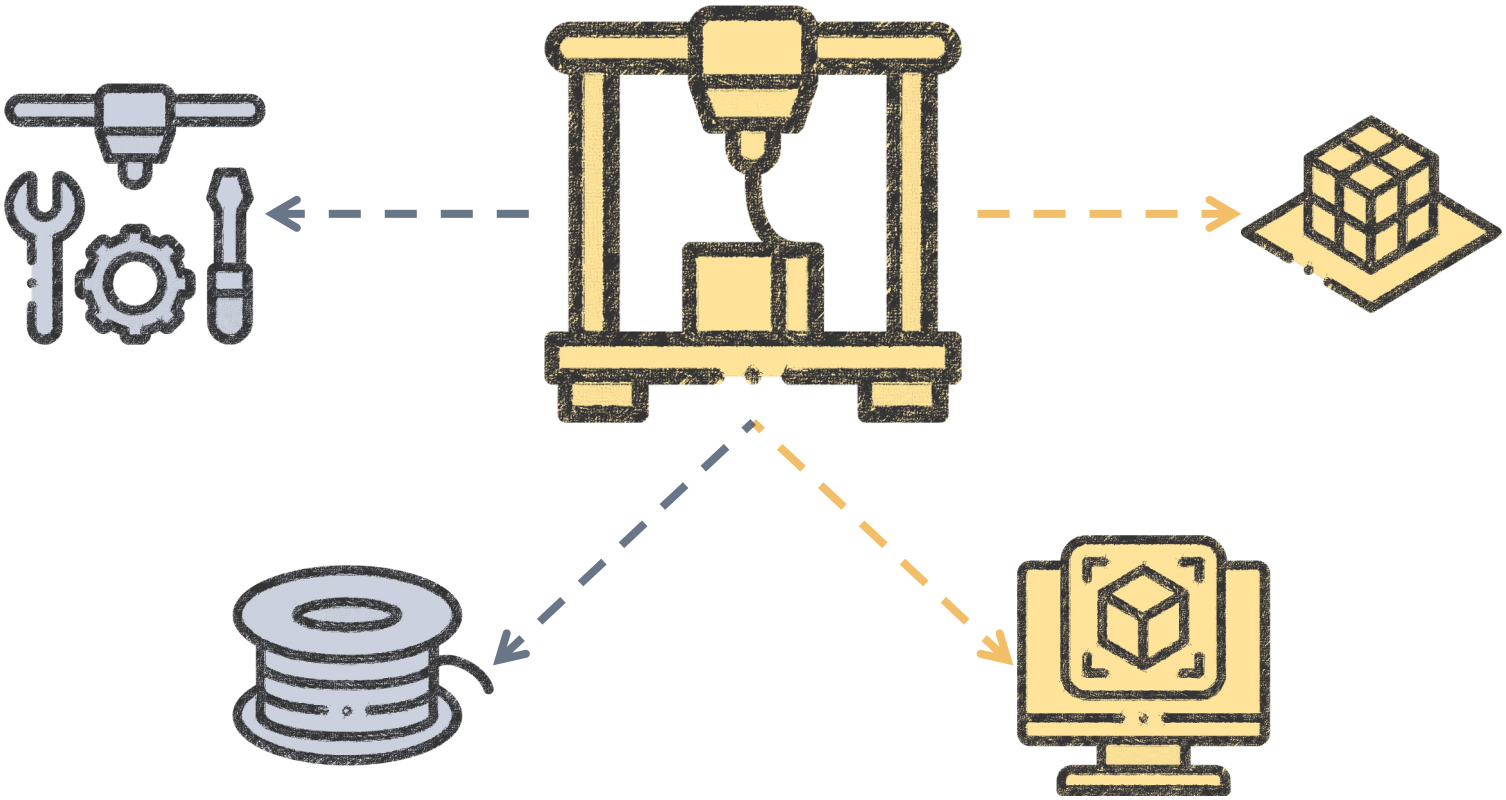
UNIVERSITÄT DES SAARLANDES

SIC Saarland Informatics Campus

3D-Druck

- Herstellung von 3D-Objekten in kurzer Zeit mit zumeist geringem Kostenaufwand
- Seit einigen Jahren immer größer werdendes Interesse
  - Sowohl im Hobbybereich
  - als auch im Bereich der Didaktik
- In der Schule:
  - Herstellung von Arbeits- und Anschauungsmaterial
  - 3D-Druck, 3D-Drucker und 3D-Design als Lerngegenstand
- Dennoch: Eher spärlicher Einsatz in Schulen
  - Kostengründe? Ab ca. 300€ für sofort einsatzfähige Geräte

Was erwartet Sie heute?

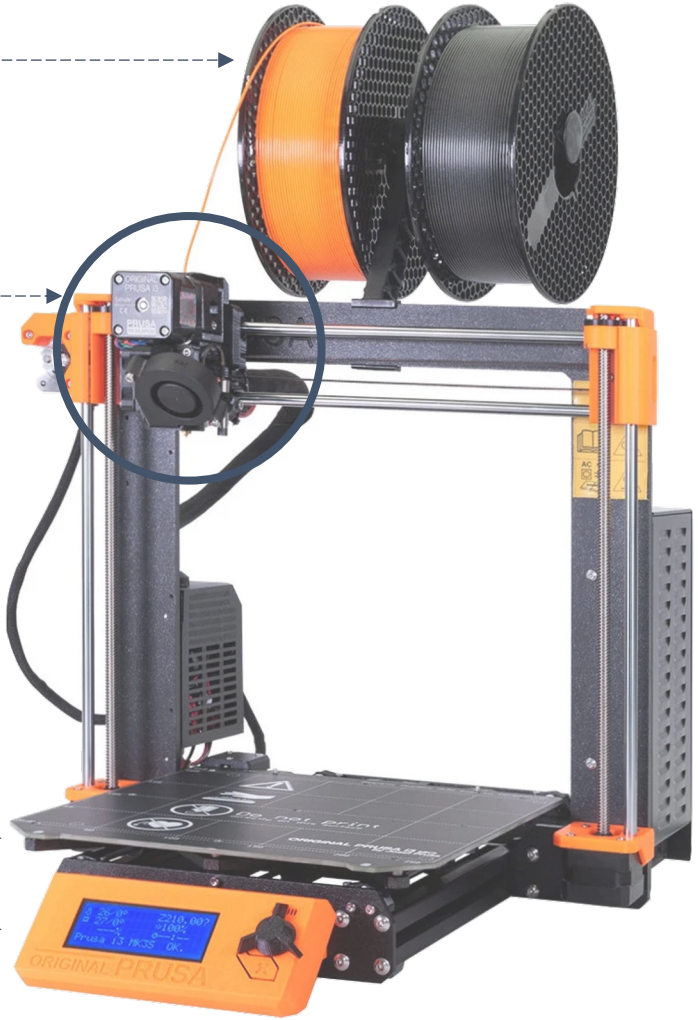


Filament

Extruder

Beheiztes Druckbett

Steuerung



FDM 3D-Drucker

Fused **D**eposition **M**odeling

Bild: https://www.prusa3d.com

Extruder

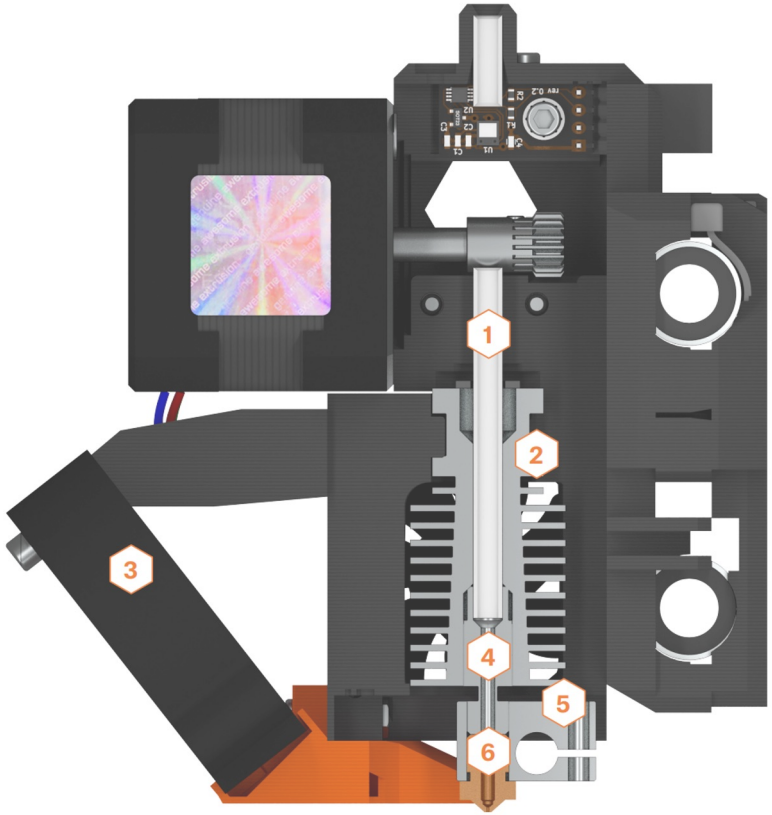


Bild: https://www.inf-schule.de/projekte/3d_druck

FDM Druckverfahren

Schicht für Schicht

FDM-Druckverfahren und viele weitere 3D-Drucktechnologien arbeiten schichtweise.

- Schichthöhen üblicherweise zwischen 0.1 und 0.3mm (bei einem Düsendurchmesser von 0.4mm)
- Digitales 3D-Objekt muss *gesliced* werden.

Bild: https://help.prusa3d.com

Filament

Vergleich verschiedener Materialien

PLA

- ca. 210 °C Düsentemperatur
- einfach zu drucken
- günstig
- spröde
- formstabil bis ca. 50 °C
- hergestellt aus Maisstärke → Nachhaltigkeit?

PETG

- ca. 230 °C Düsentemperatur
- einfach zu drucken (Heizbett erforderlich)
- etwas teurer als PLA
- dafür bessere mechanische Eigenschaften
- „lebensmittelecht“
- kann aus PET-Flaschen selbst recycelt werden → Projekt-unterricht, Nachhaltigkeit

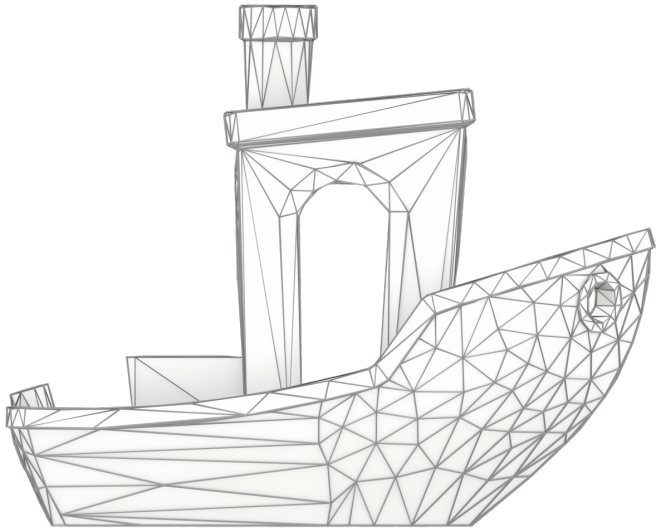
ABS

- ca. 250 °C Düsentemperatur
- eher schwierig zu drucken (*Warping*, *Layer Separation*)
- relativ günstig
- sehr gute mechanische Eigenschaften
- strenger Geruch während des Druckens (Styrol)
- Layerlinien können mit Aceton geglättet werden

3D-Design

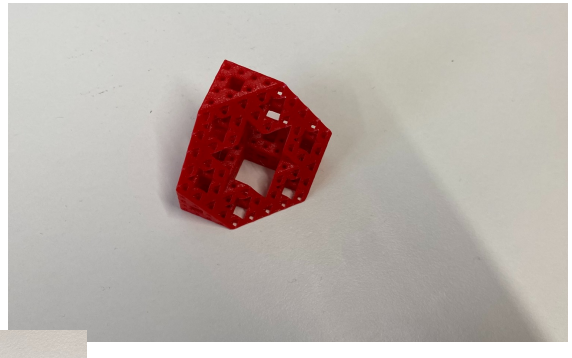
Beispielsoftware:

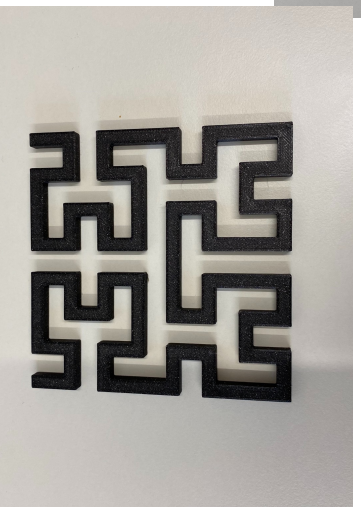
- Tinkercad
- SketchUp
- Fusion 360
- **OpenSCAD**
- GeoGebra
- …



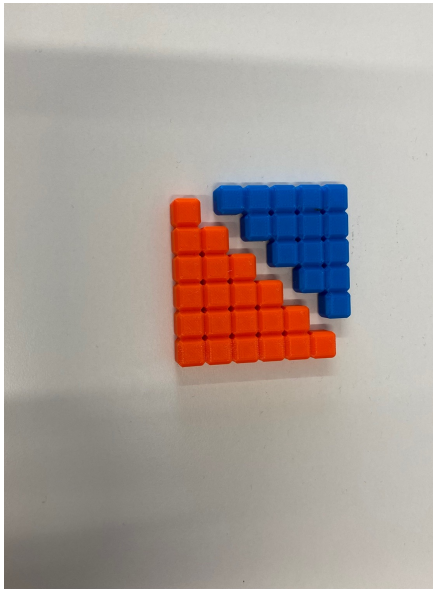
STL-Format

Rekursion





Turtle-Grafiken
Grammatiken

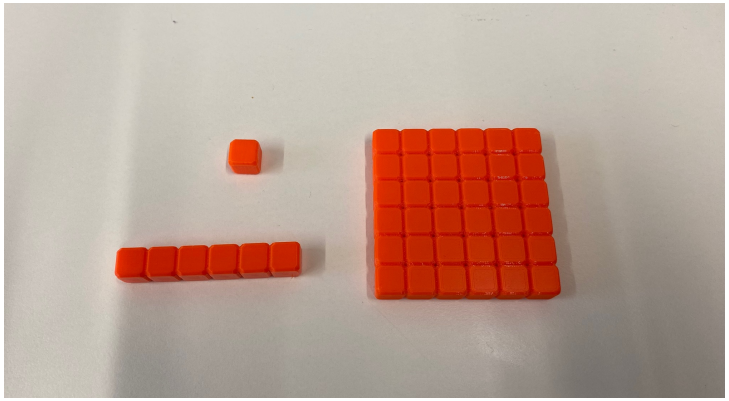


3D-Druck

Beispiele aus der Mathematik

„Informatischer“ Inhalt?

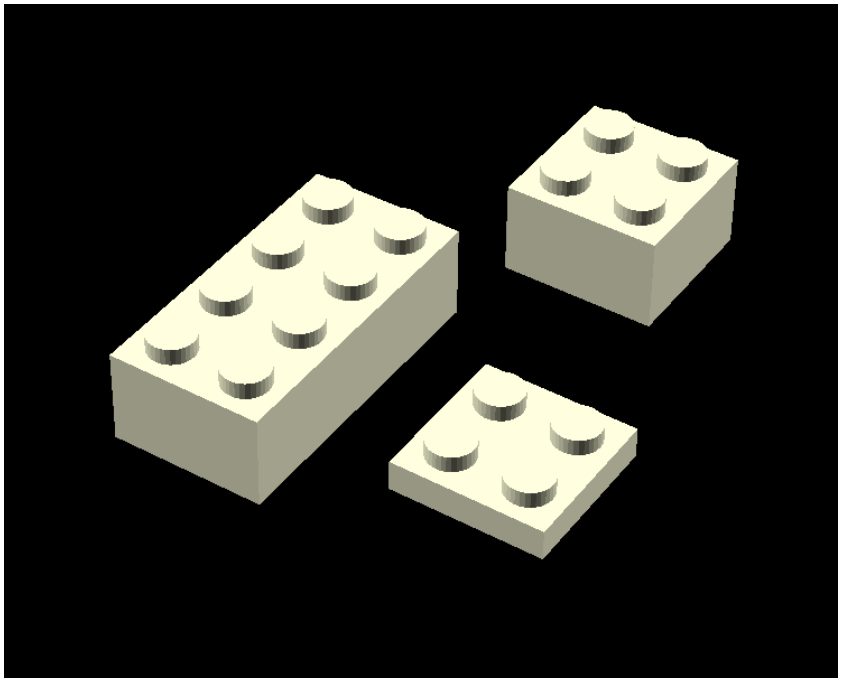
Kontrollstruktur:
Schleife



Modellieren mit OpenSCAD

Beispiel

Klemmbausteine



Progammiertechniken:

- Lineare Strukturen
- Modularisierung
- „Variablen“ / Parameter
- Schleifen, Verzweigungen
- Funktionsliterale

Speziell CAD-Konzepte:

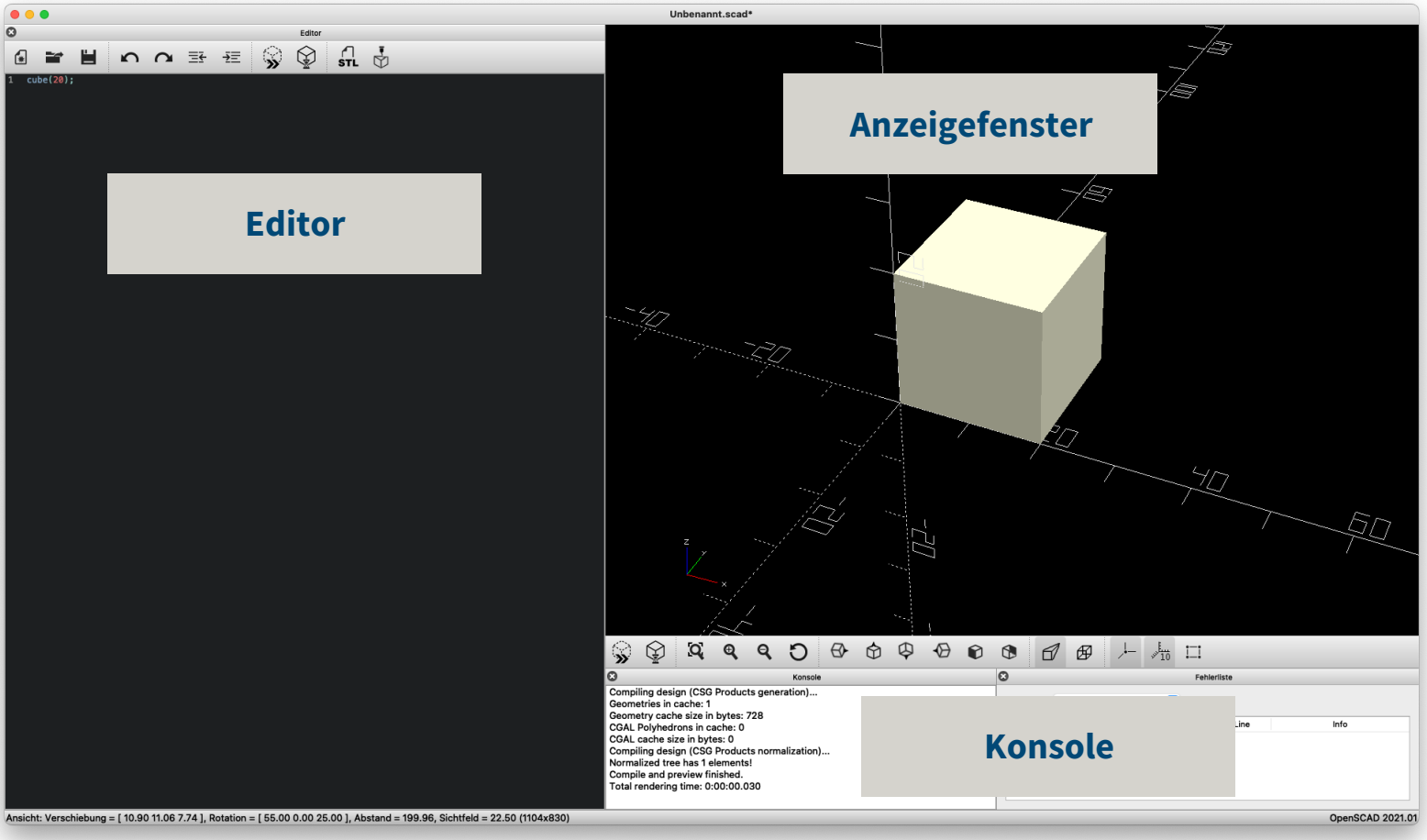
- Transformationen
- Boolesche Operationen

OpenSCAD

- **CAD: C**omputer **A**ided **D**esign
- **Funktionale** Programmiersprache zur **Beschreibung** von 3D-Objekten
- Entwicklungsumgebung ist **kostenlos** verfügbar (macOS, Windows, Linux)

Benutzeroberfläche

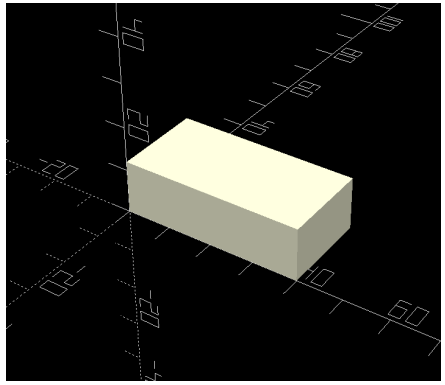
OpenSCAD



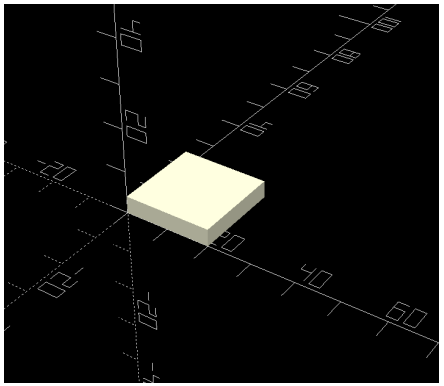
Modellieren mit OpenSCAD

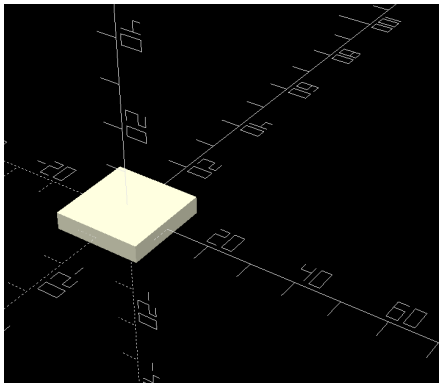
Grundelemente

- **cube**([40, 20, 12]);

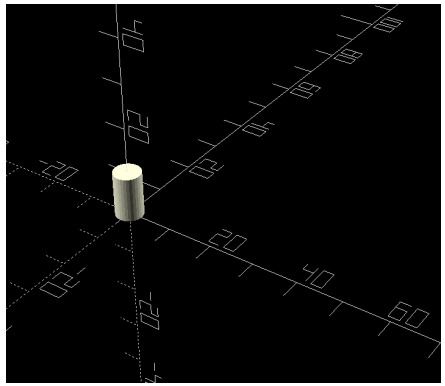


- **cube**([20, 20, 4], center = false);
  cube([20, 20, 4], center = true);





- **cylinder**([h = 2, d = 6]);



CAD-Konzepte

Transformationen

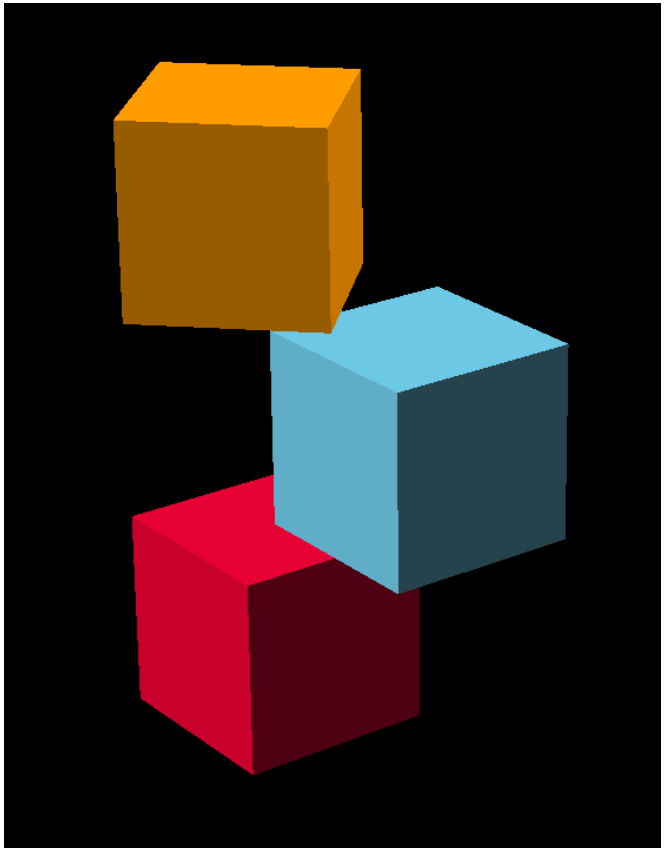
- `color("Crimson") cube(20);` Semikolon beendet Beschreibung

Translation

-
```
translate([10, 10, 20])
    color("Skyblue") cube(20);
```

Rotation

-
```
translate([0, 0, 40])
    rotate([0, 0, 45])
    color("Orange") cube(20);
```



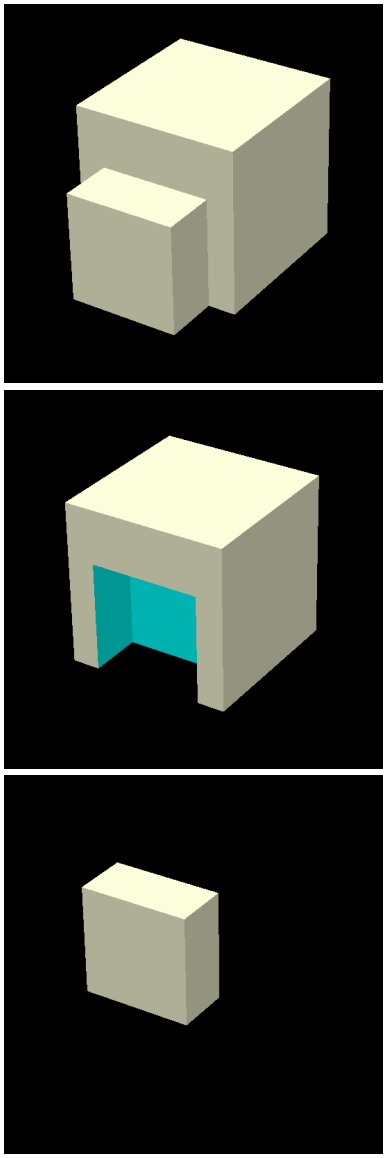
CAD-Konzepte

Boole‘sche Operationen

-
```
union() {
    translate([10, -5, 0]) cube(30);
    cube(20);
}
```

-
```
difference() {
    translate([10, -5, 0]) cube(30);
    cube(20);
}
```

-
```
intersection() {
    translate([10, -5, 0]) cube(30);
    cube(20);
}
```



Modellieren mit OpenSCAD

Variablen

Definition von Parametern:

```
toleranz = 0.1;


rastermass = 10;
noppe_durchmesser = 6 - toleranz;
noppe_hoehe = 2;
```

„Variablen“belegung kann nicht geändert werden!

→ Verhinderung von sog. Seiteneffekten

Modellieren mit OpenSCAD

Funktionen

Funktionen zur Berechnung der Steinabmessungen:

```
vert = function (anzahl_steine) noppe_raster * 1.2 * anzahl_steine;

hor = function (anzahl_noppen) anzahl_noppen * rastermass;
```

Funktionen (in Form von Literalen/Lambda-Ausdrücken) können als Parameter/Argument übergeben werden.

Modellieren mit OpenSCAD

Module

```
module basis(laenge, breite, hoehe){
    difference(){
        cube([hor(laenge), hor(breite), vert(hoehe)]);
        // TODO
    }
}
```

Module können andere Module beeinflussen („Vererbung“) → vgl. z. B. `color()`:

```
module position(x, y, z){
    // TODO
    children();
}
```

Modellieren mit OpenSCAD

Schleifen und Bedingungen

for-Schleife:

```
for (a = [0 : 3]){
    echo(a);
}
```

Ausgabe:

```
0
1
2
3
```

Reguläres If-Statement:

```
if (true) {
    cube(20);
} else {
    cylinder(h = 10, d = 20);
}
```

Conditional ?:

```
a = test ? TrueValue : FalseValue;
```

Modellieren mit OpenSCAD

Befehlsübersicht

- OpenSCAD *Cheat Sheet*

Syntax

```
var = value;
var = cond ? value_if_true : value_if_false;
var = function (x) x + x;
module name(…) { … }
name();
function name(…) = …
name();
include <….scad>
use <….scad>
```

Constants

undef	undefined value
PI	mathematical constant π (~3.14159)

Operators

n + m	Addition
n - m	Subtraction
n * m	Multiplication
n / m	Division
n % m	Modulo
n ^ m	Exponentiation
n < m	Less Than
n <= m	Less or Equal
b == c	Equal
b != c	Not Equal
n >= m	Greater or Equal
n > m	Greater Than
b && c	Logical And
b || c	Logical Or
!b	Negation

Special variables

$fa	minimum angle
$fs	minimum size
$fn	number of fragments
$t	animation step
$vpr	viewport rotation angles in degrees
$vpt	viewport translation
$vpd	viewport camera distance
$vpf	viewport camera field of view
$children	number of module children
$preview	true in F5 preview, false for F6

Modifier Characters

*	disable
!	show only
#	highlight / debug
%	transparent / background

2D

```
circle(radius | d=diameter)
square(size,center)
square([width,height],center)
polygon([points])
polygon([points],[paths])
text(t, size, font,
     halign, valign, spacing,
     direction, language, script)
import("….ext", convexity)
projection(cut)
```

3D

```
sphere(radius | d=diameter)
cube(size, center)
cube([width,depth,height], center)
cylinder(h,r|d,center)
cylinder(h,r1|d1,r2|d2,center)
polyhedron(points, faces, convexity)
import("….ext", convexity)
linear_extrude(height,center,convexity,twist,slices)
rotate_extrude(angle,convexity)
surface(file = "….ext",center,convexity)
```

Transformations

```
translate([x,y,z])
rotate([x,y,z])
rotate(a, [x,y,z])
scale([x,y,z])
resize([x,y,z],auto,convexity)
mirror([x,y,z])
multmatrix(m)
color("colorname",alpha)
color("#hexvalue")
color([r,g,b,a])
offset(r|delta,chamfer)
hull()
minkowski(convexity)
```

Lists

```
list = […, …, …];  create a list
var = list[2];  index a list (from 0)
var = list.z;  dot notation indexing (x/y/z)
```

Boolean operations

```
union()
difference()
intersection()
```

List Comprehensions

```
Generate [ for (i = range|list) i ]
Generate [ for (init;condition;next) i ]
Flatten [ each i ]
Conditions [ for (i = …) if (condition(i)) i ]
Conditions [ for (i = …) if (condition(i)) x else y ]
Assignments [ for (i = …) let (assignments) a ]
```

Flow Control

```
for (i = [start:end]) { … }
for (i = [start:step:end]) { … }
for (i = […,…,…]) { … }
for (i = …, j = …, …) { … }
intersection_for(i = [start:end]) { … }
intersection_for(i = [start:step:end]) { … }
intersection_for(i = […,…,…]) { … }
if (…) { … }
let (…) { … }
```

Type test functions

```
is_undef
is_bool
is_num
is_string
is_list
is_function
```

Other

```
echo(…)
render(convexity)
children([idx])
assert(condition, message)
assign (…) { … }
```

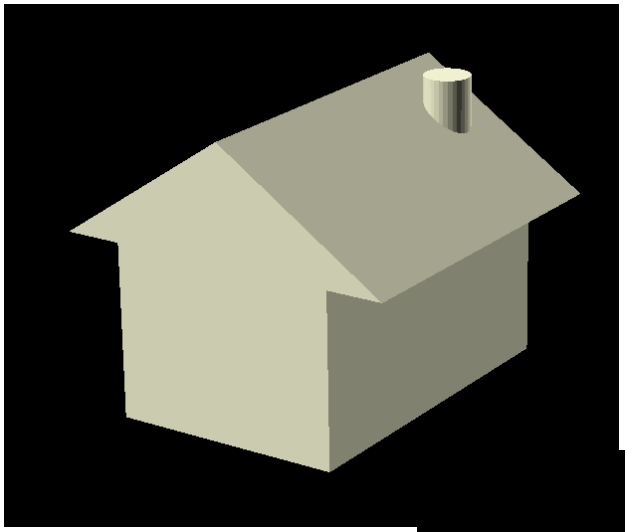
Functions

```
concat
lookup
str
chr
ord
search
version
version_num
parent_module(idx)
```

Mathematical

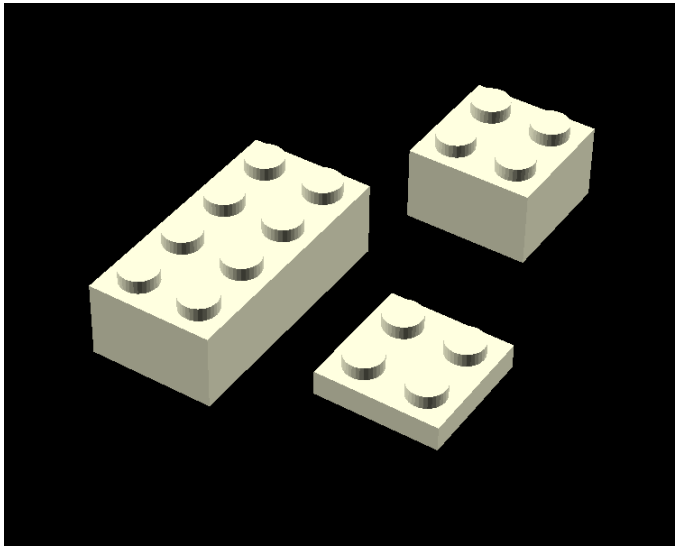
```
abs
sign
sin
cos
tan
acos
asin
atan
atan2
floor
round
ceil
ln
len
let
log
pow
sqrt
exp
rands
min
max
norm
cross
```

SCAN ME



Vorschläge/Anregungen:

- Modellierung einfacher Objekte: Dreiecksprisma, Haus, Roboter, …
- Modellierung von Klemmbausteinen ***https://kurzelinks.de/vorlage_tdiu***
- Eigene Anwendungsideen (auch fächerübergreifend)?



Sie sind an der Reihe!

Abschluss & Diskussion

Fazit

- Sichtbar-machen von Programmstrukturen, wie z. B. Schleifen und Rekursion
- OpenSCAD erlaubt dabei auch Nutzung funktionaler Konzepte
- 3D-Druck ermöglicht Herstellung enaktiv nutzbarer Materialien

Vorschläge zur Diskussion

- 3D-Druck und/oder 3D-Design im Informatikunterricht
- Stellenwert des funktionalen Programmierparadigmas
- Bedeutung für die Sekundarstufe I