13. Module und Funktionen

In OpenSCAD kann man Konstruktionsschritte zu Modulen zusammenfassen und sie über den Namen benutzen. Zur besseren Lesbarkeit kann man einzelne Berechnungen als Funktionen definieren. An beide Konstrukte kann man Parameter übergeben.

```
module name_des_moduls(parameter_1, parameter_2)
{
      folge
            von
                  anweisungen();
}

function pythargoras(a, b)=sqrt(pow(a, 2) + pow(b, 2));

c=pythargoras(10, 15);
```

13.1 Aufgabe 15

Ergänze die Lösung zur Aufgabe 14 so, dass man beliebige Alu-Stränge erzeugen kann, wenn man die Länge des Strangs übergibt. Der Aufruf `alu_profil_40_x_40(100);` soll einen Strang von 100mm erzeugen.

Verfahre ebenso mit der Lösung zur Aufgabe 12. Das Modul soll als `wuerfelverbinder_40_x_40();` aufgerufen werden.

13.2 Kinder

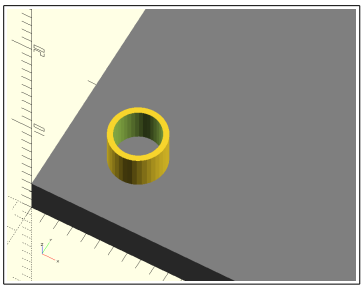
Module können auch agieren, als wenn sie Anweisungen wie z.B. `rotate()` oder `translate()` wären. Sie erzeugen selbst keine Objekte, manipulieren aber die Nachfolgenden. Innerhalb des Moduls steht das Objekt als `children()` zur Verfügung. Häufige Anwendungen sind Konstruktionen mit Löchern. Jedes Mal wenn so eine Teilkonstruktion dem Größeren hinzugefügt wird, stellt man fest, dass vor allem Löcher nicht mehr durchgängig sind, da das Loch nur innerhalb der Teilkonstruktion definiert ist.

```
module abstandshalter_m3(hoehe)
{
      difference()
      {
            cylinder(d=4, h=hoehe, $fn=20);
            translate([0, 0, -0.5])
                  cylinder(d=3.2, h=hoehe + 1, $fn=20);
      }
}

platte_x=100;
platte_y=50;
platte_z=3;

cube([platte_x, platte_y, platte_z]); // Platte
for(x=[5, platte_x - 5], y=[5, platte_y - 5])
{
      translate([x, y, platte_z - 0.5])
            abstandshalter_m3(3.5);

}
```



Obwohl der Abstandshalter vollständig durch gebohrt ist, enthält die Platte kein Loch. Man müsste nun innerhalb der for-Schleife noch einmal eine Lochkonstruktion definieren. Besser ist es, die Positionierung in ein Modul zu tun und mittels `children()` den Abstandshalter bzw. das Durchgangsloch zu konstruieren:

```
module durchgangsloch_m3(laenge)
{
	translate([0,0,-3.5])
		cylinder(d=3.2, h=laenge + 0.5);
}

module abstandshalter_m3()
{
	difference()
	{
		cylinder(d=4, h=3);
		translate([0, 0, -0.5])
			cylinder(d=3.2, h=3 + 1);
	}
}

module abstandshalter_positionen()
{
	for(x=[5, platte_x - 5], y=[5, platte_y - 5])
	{
		translate([x, y, platte_z - 0.5])
			children();
	}
}

$fn=40;
platte_x=100;
platte_y=50;
platte_z=3;

difference()
{
	union()
	{
		cube([platte_x, platte_y, platte_z]); // Platte

		abstandshalter_positionen()
			abstandshalter_m3();
	}
	abstandshalter_positionen()
		durchgangsloch_m3(6);
}
```

Weitere Anwendungen von `children()` kannst du hier nachlesen.

14. Bibliotheken

OpenSCAD kann bestehende Konstruktionsschritte aus anderen Dateien einbinden. Dies geschieht mittels `use <Dateiname>` bzw. `include <Dateiname>`. Der Unterschied zwischen beiden ist, dass bei `use` lediglich die Module und Funktionen importiert, etwaige Anweisungen aber nicht ausgeführt werden. Ein `include` würde auch die enthaltenen Anweisungen ausführen. OpenSCAD kennt 3 Speicherorte für Bibliotheksdateien:

- Installations-Bibliotheken werden zur Programminstallation mitgeliefert, befinden sich im Ordner libraries des Installationspfades und sind für alle Benutzer des Systems verfügbar. Dort befindet sich momentan nur die MCAD-Bibliothek. Ein `include <MCAD/involute_gears.scad>` bindet z.B. die Bibliothek für Zahnräder mit Evolventenverzahnung ein.
- Persönliche Bibliotheken liegen im Ordner
  - (Windows Systeme) eigene Dokumente\OpenSCAD\libraries
  - (Linux Systeme) $HOME/.local/share/OpenSCAD/libraries

- (Mac OS X Systeme) $HOME/Documents/OpenSCAD/libraries
- Zum Erzeugen metrischer Gewinde kann man diese Bibliothek verwenden. Nach dem Abspeichern in obigem Ordner kann man die Bibliothek mit `include <threads.scad>` einbinden.

- OpenSCAD sucht eine Bibliothek aber auch im aktuellen Ordner einer geöffneten Konstruktionsdatei.

14.1 Aufgabe 16

Erzeuge aus der Lösung zur Aufgabe 15 eine Bibliothek.

15. Was noch fehlt

Auf den vorigen 12 Seiten habe ich nicht alle Möglichkeiten von OpenSCAD aufgeführt. Einige sind schon ziemlich exotisch wie z.B. `multimatrix` oder `offset` und können durch Studium der originalen Dokumentation erlernt werden. Ich habe z.B. versucht, auf die Radius-Schreibweise zu verzichten, welche man alternativ zu den Durchmessern verwenden kann. Eine Anweisung `cylinder(r1=50, r2=10, h=50);` erzeugt den gleichen Kegelstumpf wie `cylinder(d1=100, d2=20, h=50);` - in der Praxis wirst Du aber sehr selten Radien begegnen - man kann Sie schlicht nicht an einem Objekt messen. Mit einem Meßschieber misst man immer die Durchmesser von Rundungen.

Ein paar wichtige abschließende Dinge möchte ich dennoch hier bringen:

- Will man etwas auf der Konsole ausgeben, nimmt man `echo("Hier dein Text.", abstand, "mm fehlen noch.");` Damit kann man Werte ausgeben lassen, um Fehler zu finden oder sog. BOMs (bill of materials = Teileliste) erstellen.
- Sehr oft kommt es vor, dass man bereits fertige Dateien manipulieren möchte. Dateien im .stl Format können mit `import("dateiname.stl");` importiert werden. Anschließend kann das Objekt mit den bekannten Manipulationen bearbeitet und verändert werden.
- Ebenso ist es möglich .dxf Dateien zu importieren. Ein zusätzlicher Parameter layer= legt fest, welchen Teil der .dxf Datei man importieren möchte. Die importierten Teile sind 2D-Objekte!
- OpenSCAD kennt drei verschiedene for-Schleifen
  - `for(variable=[wert_1, wert_2, … wert_n]) { anweisung(en); }` Die Variable erhält pro Schleifendurchlauf nacheinander jeden Wert der Werteliste.
  - `for(variable=[anfangswert : inkrement : endwert]) { anweisung(en); }` Die Variable erhält pro Schleifendurchlauf beginnend mit dem Anfangswert jeden durch den Inkrement gebildeten Wert bis maximal zum Endwert. Je nach den gewählten Werten muss der Endwert nicht unbedingt erreicht werden. Wird der Inkrement weggelassen, wird 1 als Inkrement angenommen. Als Inkrement sind auch negative Werte erlaubt.
  - `for(variable = [ [vektor_1], [vektor_2], … [vektor_n] ]) { anweisung(en); }` Die Variable erhält pro Schleifendurchlauf nacheinander jeden Vektor der Vektorliste.
  - For-Schleifen dürfen auch ineinander geschachtelt werden. Entweder klassisch durch ineinander schreiben oder so: `for(a=[1, 2, 3], b=[4, 5]) { echo(„a=“, a, „b=“, b); }` Diese Schleife wird auf der Konsole den Text a=1 b=4, a=1 b=5, a=2 b=4 a=2 b=5, a=3 b=4, a=3 b=5 ausgeben.
- Mit der Fallunterscheidung `if(Bedingung) { Anweisungen } else { Anweisungen }` können Konstruktionen von Bedingungen abhängig werden. An Bedingungen gibt es
  - größer (>),
  - größer oder gleich (>=),
  - kleiner (<),
  - kleiner oder gleich (<=),

  - gleich (==) oder
  - ungleich (!=).
  - Anweisungen mit Wertzuweisungen sind hier nicht zugelassen! Dafür gibt es den „Conditional Operator“

- Der Conditional Operator hilft aus, da Wertzuweisungen innerhalb einer if-Fallunterscheidung nicht zugelassen sind: `name = Bedingung ? Wert für wahr : Wert für falsch;`

  Beispiel:

```
a=5;
b=10;
c = a > b ? „ist größer“ : „ist kleiner“;
```

- An Funktionen gibt es u.a.
  - die trigonometrische Funktionen `sin()`, `cos()`, `tan()` und deren Umkehrfunktionen `asin()`, `acos()` und `atan()`
  - `pow(basis, exponent)` liefert die Potenz
  - `sqrt()` liefert die Quadratwurzel. Andere Wurzelberechnungen wie etwa die dritte Wurzel müssen mit `pow(1/3, wert)` gebildet werden.
- Weitere Funktionen
  - `lookup()` sucht in einer Liste nach einem Treffer und liefert einen linear interpolierten Wert, wenn es keinen exakten Treffer gibt.
  - `search()` sucht in einer Liste nach einem oder mehreren Treffern und liefert keinen Wert, wenn es keinen exakten Treffer gibt.

Es gibt noch eine handvoll weitere Anweisungen und Funktionen in OpenSCAD. Diese werden zu einem späteren Zeitpunkt behandelt, wenn ich passende Beispiele dafür habe. Damit ist das Kapitel „01 Einführung“ abgeschlossen. Es folgen noch ein paar Aufgaben, die das bisher vermittelte Wissen vertiefen sollen.

15.1 Aufgabe 17

Bearbeite die Lösung zur Aufgabe 16 so, dass jedes mal, wenn eines der Module aufgerufen wird, an der Konsole der Text

- „Alu Strangprofil 40mm x 40mm x [übergebene Länge]mm“

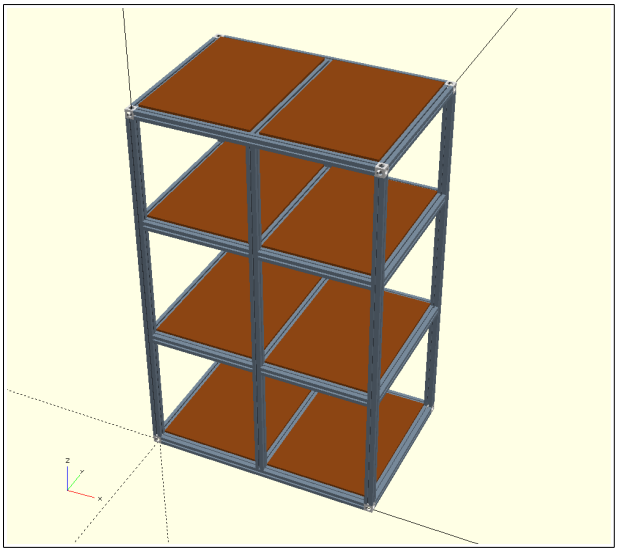
bzw.

- „Wuerfelverbinder 40mm x 40mm“

ausgegeben wird. Dies wird später für eine Einkaufsliste benötigt.

Der 2D-Teil im Modul `alu_profil_40_x_40` ist mit den `rotate()` bzw. `mirror()` ziemlich lang geworden. Überarbeite diesen Teil und benutze eine for-Schleife.

15.2 Aufgabe 18



Das 3D-Drucker Rack besteht aus 6 Fußprofilen (senkrecht), 12 Profilen für die Tiefe (waagerecht nach hinten), 12 Profilen für die Breite (waagerecht zur Seite, je 4 lange für oben und unten sowie 8 kurze) und 8 Würfelverbinder (je 4 unten und oben). Damit ergeben sich unten 2 Fächer für Filamente, in der Mitte 4 Fächer für Drucker und oben nochmal 2 Möglichkeiten für diverses Zubehör. Die Fachböden sind jeweils 0,5m breit und 0,7m tief und sollen auf noch zu konstruierende Winkelstücke lagern.

Konstruiere ein Rack aus der in Aufgabe 16 erstellten

Bibliothek in den Maßen

- 1,77m Höhe
- 1,12m Breite
- 0,78m Tiefe

Zur Positionierung der Bauteile eignen sich for-Schleifen. Für die Würfelverbinder könnte das z.B. so aussehen:

```
wuerfel_delta_x=1120;
wuerfel_delta_y=700;
wuerfel_delta_z=1690;

for(x=[0 : wuerfel_delta_x : 1120],
    y=[0 : wuerfel_delta_y : 780],
    z=[0 : wuerfel_delta_z : 1770])
{
      translate([x, y, z])
            wuerfelverbinder_40_x_40();
}
```

Für die senkrechten Profile benötigt man nur eine 2-fach geschachtelte for-Schleife (x und y). Die 12 Profile für die Tiefe bekommt man wieder über eine 3-fache for-Schleife. Für die Breite braucht es eigene Schleifen für die 4 längeren Stücke (je 2 oben und unten) sowie für die 8 kurzen Stücke im Mittelteil.

Bonusrunde: Konstruiere die Fachböden mit in das Rack.

Bonusrunde 2: Berücksichtige in der BOM auch die Schrauben für die Würfelverbinder.

15.3 fensterloser Betrieb

Beachte die Ausgaben von Aufgabe 18 auf der Konsole. Geeignete Umwandlungsschritte mit den unter Linux gebräuchlichen Textfilter-Werkzeugen wie grep, sed, sort und uniq können aus den Ausgaben eine verwendbare Einkaufsliste erzeugen. Für Linux sind diese Tools bereits installiert. Windows-Anwender können die Tools von hier beziehen und müssen anschließend die Dateien grep.exe, sed.exe, sort.exe und uniq.exe in den Ordner mit der .scad Datei kopieren.

Um die Ausgaben der Konsole in eine Textdatei zu bekommen, kann man entweder die Zwischenablage benutzen und mittels copy&paste den Inhalt in einem Texteditor fallen lassen, oder man startet OpenSCAD über die „Eingabeaufforderung“. Richtig komfortabel wird es, wenn man die notwendigen Befehle in eine ausführbare Scriptdatei steckt. Für Windows könnte das so aussehen:

```
@echo off
set eingabe="Aufgabe 18 - Bonus 2.scad"
set ausgabe=Aufgabe_18_Bonus_2.png
set bomdatei=bom.txt
"c:\Program Files\OpenSCAD\openscad.exe" %eingabe% -o %ausgabe% 2>bom_tmp.txt
grep ECHO bom_tmp.txt |^
sort |^
uniq -c |^
sed -e "s/ECHO: \"//g" -e "s/\",//g" -e "s/, \"//g" -e "s/\"//g" > %bomdatei%
del bom_tmp.txt
cls
more %bomdatei%
```

Dieses Script startet OpenSCAD (ohne Fenster), berechnet die Vorschau der Eingabedatei, speichert ein Bild der Vorschau ab und schreibt die Ausgabe der Konsole (Dateideskriptor 2 / stderr) in eine temporäre Datei. Anschließend wird diese Datei durch die Filter grep, sort, uniq und sed so umgebaut, dass am Ende die Einkaufsliste in der bomdatei landet:

```
      4     Alu Strangprofil 40mm x 40mm x 1040mm
      6     Alu Strangprofil 40mm x 40mm x 1690mm
     12     Alu Strangprofil 40mm x 40mm x 500mm
     12     Alu Strangprofil 40mm x 40mm x 700mm
```

```
     24     Schraube M8x25
      8     Wuerfelverbinder 40mm x 40mm
```

Weitere Parameter für den „fensterlosen“ Betrieb von OpenSCAD findest du hier.

15.4 Aufgabe 19

Die Würfelverbinder sollen in der Zeichnung so orientiert werden, dass man die Schrauben auch hinein bekommt. D.h. die größeren Öffnungen für die Schraubenköpfe sollen nach außen zeigen. Ändere das Modul `wuerfelverbinder_40_x_40` so ab, dass man dem Aufruf drei Zeichen mitgeben kann. Je eines für die Orientierung der Löcher einer der drei Achsen. Ein Aufruf von `wuerfelverbinder_40_x_40("uhl")` soll bedeuten, dass die Schrauben von unten, hinten und von links eingeführt werden sollen. Berücksichtige natürlich auch die anderen Fälle (oben, vorn und rechts).

```
if (orientierung[0] == "o")
{
      // oben
} else
{
      // unten
}
if (orientierung[1] == "v")
{
      // vorn
} else
{
      // hinten
}
```

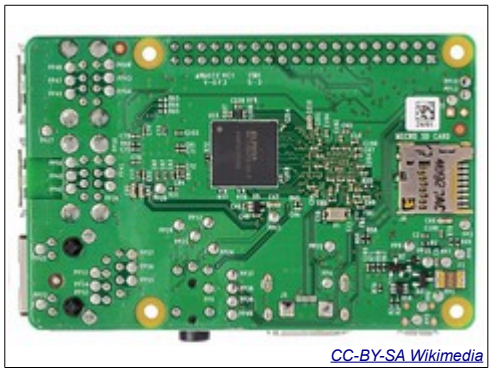
15.5 Aufgabe 20

Ein Raspberry Pi B Modell 2 soll in ein handelsübliches 19“ Rack eingebaut werden. Konstruiere einen 1 HE Montagewinkel, der auf der linken Seite des Racks mit 2 M6x20 Schrauben festgeschraubt werden kann. Da kein verfügbarer 3D Drucker in der Lage ist, einen 19“ (=482,6mm) langen Rackwinkel zu drucken und die aufzunehmenden Gewichtskräfte gering sind, genügt es, die Verschraubung nur links zu konstruieren.

CC-BY-SA Wikimedia

Orientiere den Raspberry so, dass der Netzwerkanschluss zugänglich ist. Die anderen Anschlüsse des Raspberry sind nicht in Verwendung.

Die nötigen Abmessungen des Racks und Raspberry kannst du aus dem Intrernet beschaffen. Fehlende Angaben dürfen geschätzt, oder falls vorhanden, auch direkt abgemessen werden. Achte darauf, dass kein Bauteil über die HE hinaus stehen darf. Da der Raspberry unter der Platine den Kartenleser hat, ist eine geeignete Stützkonstruktion nötig. Es ragen auch die Enden der bedrahteten Bauteile nach unten durch die Platine.



CC-BY-SA Wikimedia