Programmierung von CAx-Systemen

David Straub

Ziel dieser Lehrveranstaltung

Die Studierenden erwerben ein fundiertes Verständnis der geometrischen **Grundlagen der 3D-Modellierung** sowie der **algorithmischen Erzeugung** und **Optimierung von CAD-Geometrien**.

Sie lernen, parametrische Modelle **mithilfe geeigneter Programmiersprachen** oder Skriptumgebungen zu erstellen, modellierungsrelevante Abläufe zu **automatisieren** und bestehende Workflows hinsichtlich Effizienz und Robustheit zu verbessern.

Zudem entwickeln sie Kompetenzen im Zusammenspiel von skriptbasierter Geometrieerzeugung und grafischen CAD-Umgebungen.

Modulhandbuch SoSe 2026, TBM 2.2

Organisatorisches

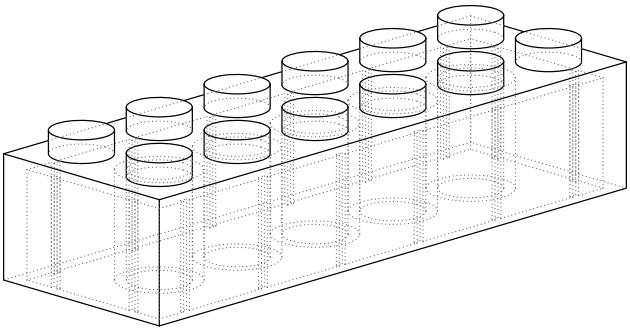
- Termine (4 SWS):
  - SU Donnerstag 9:00-10:30 Uhr, B158
  - Ü Donnerstag 10:45-12:15 Uhr, B356
- Hardware: eigenes Laptop oder Laborrechner
- Prüfung: schriftlich, 90 Minuten

Motivation

Parametrische Modellierung: warum überhaupt?

Direkte Modellierung - Geometrie wird manuell geformt (Push/Pull) - Feste Koordinaten, keine Historie

Parametrische Konstruktion - **Rezept statt Endprodukt:** Form aus Variablen und Beziehungen - **Design Intent (Konstruktionsabsicht):** Regeln definieren Geometrie - Parameter ändern → Modell berechnet sich neu - Ideal für Variantenmanagement und Produktfamilien



Parametrische Moellierung: warum Code statt Maus?

Problem: CAD-GUI-Modellierung - Keine Nachvollziehbarkeit (Wer hat was wann geändert?) - Wiederholaufgaben manuell → fehleranfällig - Variantenbildung = viele Dateien kopieren - Design-Optimierung nur durch Trial & Error

Lösung: Code-basierte CAD-Konstruktion - Versionsverwaltung (Git) - Automatisierung von Wiederholaufgaben - Algorithmische Optimierung

Versionsverwaltung

Klassische CAD-Dateien (binär): - `gehaeuse_v1.sldprt`, `gehaeuse_v2_final.sldprt`, `gehaeuse_v2_wirklich_final.sldprt` - Keine Änderungshistorie sichtbar - Merge-Konflikte unlösbar

Code-basierte CAD-Modelle (Text):

```
# Änderung nachvollziehbar in Git:
- breite = 100  # alt
```

```
+ breite = 120  # neu
```

- **Vollständige Historie** aller Änderungen
- **Branching**: Parallele Varianten entwickeln
- **Teamarbeit**: Mehrere Teammitglieder, ein Modell

Automatisierung

Szenario: 200 Schrauben in Normgrößen

```
for durchmesser in [3, 4, 5, 6, 8, 10, ...]:
    schraube = erzeuge_schraube(durchmesser, laenge)
    exportiere(f"ISO4762_M{durchmesser}.step")
```

- **Fehlerfrei**, da einmal programmiert
- **Reproduzierbar** auf Knopfdruck

Algorithmische Optimierung

```
result = minimize(
    fun=berechne_masse,
    x0=[wandstaerke, rippenabstand],
    constraints={'type': 'ineq', 'fun': lambda x: festigkeit(x) - 500}
)
# Finde minimale Masse bei Festigkeit ≥ 500 MPa
```

- **Hunderte Varianten** automatisch durchrechnen
- **Optimales Design** mathematisch finden

Beispiel: Fusionsreaktor

Link

KI-CAD

- KI-generierte Geometrie aus Textbeschreibung
- Beispielanwendung: “Reverse Engineering CAD Code from Point Clouds” arXiv:2412.14042

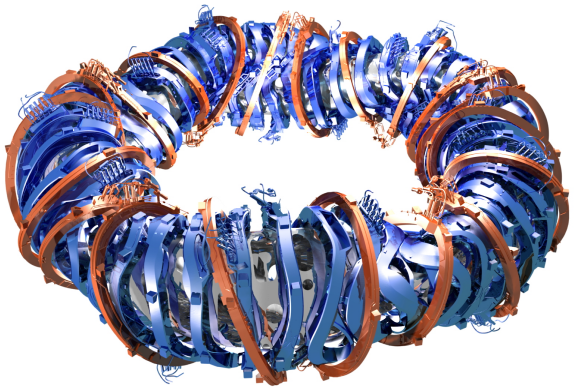
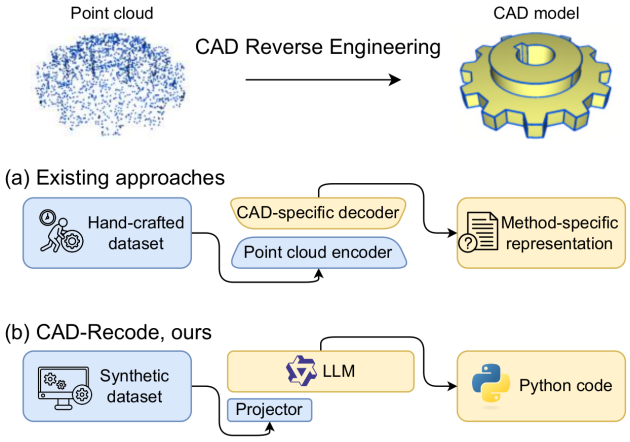


Figure 1: width:15cm



Warum Python?

- Nicht CAD-spezifisch, sondern **Allzweck-Programmiersprache**, bekannt aus Informatik 1
- Derzeit weltweit **beliebteste Programmiersprache**
- **Plattformunabhängig**: läuft auf Windows, Linux, macOS
- **Quelloffen** und kostenlos

Gliederung

1. Einführung
2. Topologie
3. Geometrie
4. Modellierungsstrategien
5. Datenaustausch

6. Simulation
7. Optimierung
8. Fertigung

Einführung

- ~~Motivation~~
- Darstellung von Geometrie in CAD-Systemen
- Geschichtlicher Exkurs: Open-Source-CAD-Programmierung
- Moderner Python-CAD-Stack

Darstellung von Geometrie in CAD-Systemen

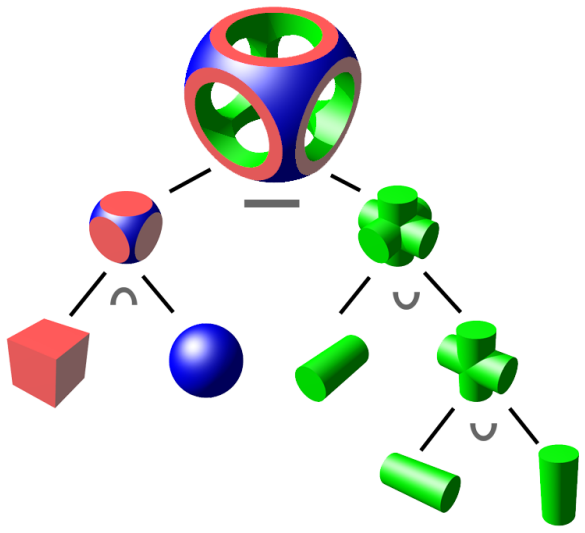
Dreidimensionale Geometrie kann in CAD-Systemen auf verschiedene Arten dargestellt werden:

- **CSG (Constructive Solid Geometry):** Volumen durch boolesche Operationen einfacher Körper
- **B-Rep (Boundary Representation):** Oberfläche definiert Volumen, Kanten definieren Flächen
- **Drahtgitter (Mesh):** Overfläche aus Polygonen, z.B. Dreiecken

CSG (Constructive Solid Geometry)

Konstruktive Festkörpergeometrie

- Modellierung durch boolesche Operationen (Union, Intersection, Difference)
- Basiert auf einfachen Grundkörpern (Würfel, Zylinder, Kugel, Kegel)
- Hierarchische Struktur als CSG-Baum
- **Vorteil**: Kompakte Darstellung, einfache Modifikation
- **Nachteil**: Begrenzte geometrische Komplexität

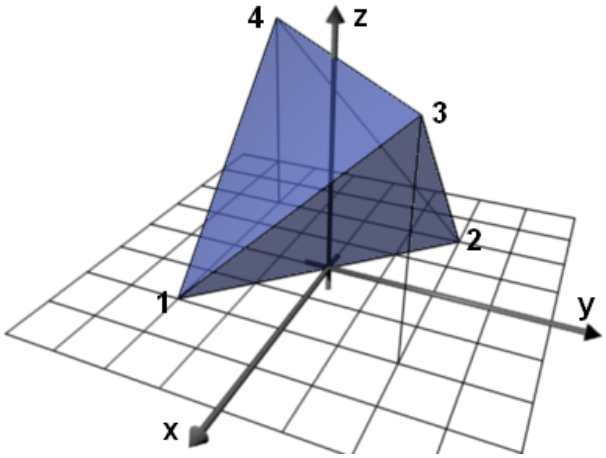


B-Rep (Boundary Representation)

Begrenzungsflächenmodell

- Geometrie durch **Oberfläche** beschrieben

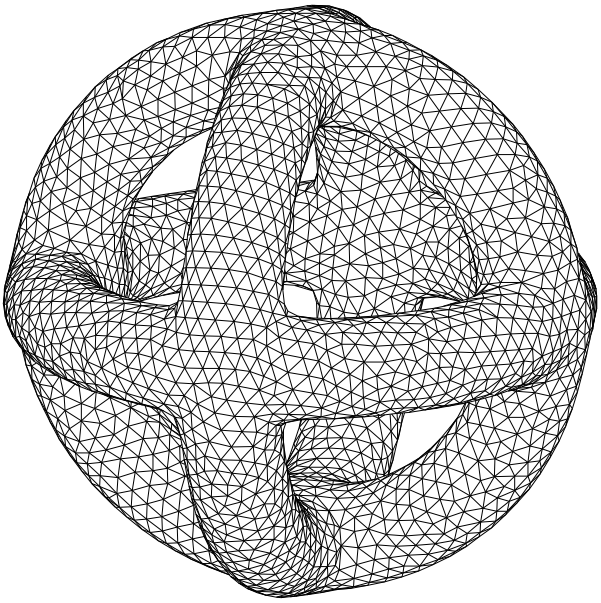
- Hierarchie: Faces (Flächen), Edges (Kanten), Vertices (Ecken)
- Verwendet Splines für gekrümmte Flächen
- **Vorteil**: Hohe Präzision, komplexe Formen möglich
- **Nachteil**: Größerer Speicherbedarf, komplexe Datenstrukturen
- Standard in professionellen CAD-Systemen



Mesh (Netz)

Polygonnetz-Darstellung

- Oberfläche aus **Dreiecken** oder **Vierecken**
- **Keine exakte Geometrie**, nur Annäherung
- **Anwendungen**: 3D-Scanning, 3D-Druck (STL), FEM, Visualisierung
- **Vorteil**: Einfache Darstellung, schnelle Visualisierung
- **Nachteil**: Keine parametrische Bearbeitung, speicherintensiv

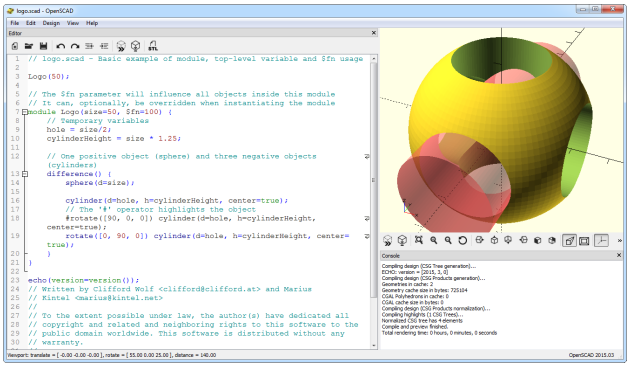


Geschichtlicher Exkurs: Open-Source-CAD-Programmierung

OpenSCAD

Erste Open-Source-CAD-Skripting-Software (2010)

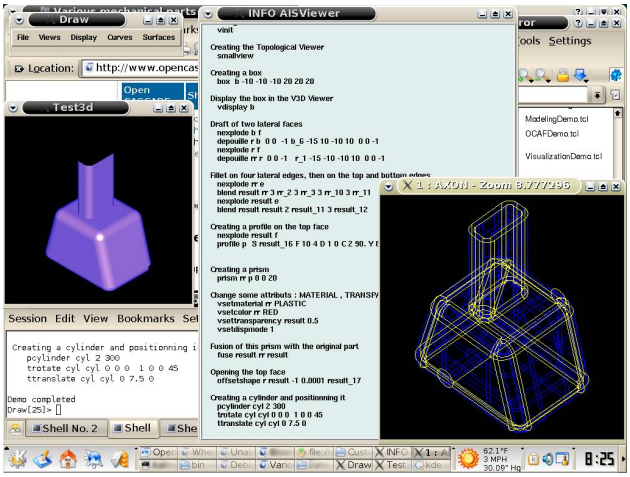
- Verwendet eine eigene Skriptsprache
- Basiert auf CSG (Constructive Solid Geometry)
- Kein Export in Standard-CAD-Formate wie STEP



Open CASCADE Technology (OCCT)

Open-Source-CAD-Kernel (1999)

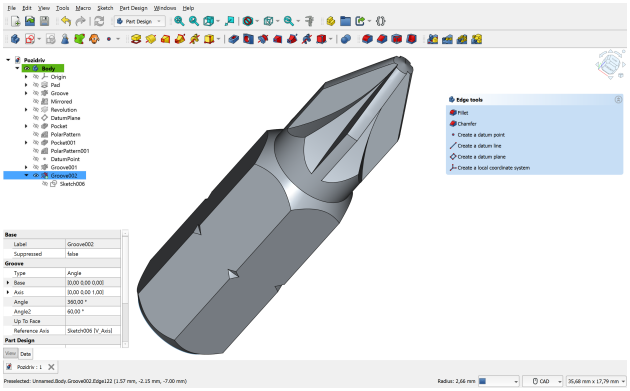
- C++-Bibliothek für **geometrische Modellierung** und **Visualisierung**
- Basiert auf B-Rep
- Grundlage für viele quelloffene und kommerzielle CAD-Systeme
- Lizenz: LGPL (frei nutzbar, auch kommerziell)



FreeCAD

Open-Source-CAD-Software (2002)

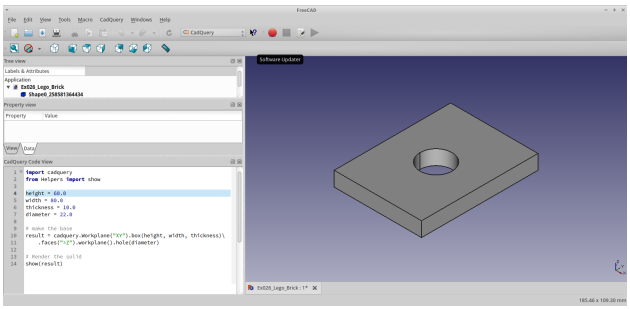
- Basiert auf **OCCT** als Geometrie-Kernel
- GUI-basierte parametrische 3D-Modellierung
- **Python-API** für Skripting und Automatisierung
- Export in STEP, STL und andere Formate
- Plattformunabhängig (Windows, Linux, macOS)



CadQuery 1.0

Python-Bibliothek für parametrisches CAD (2012)

- Erste Code-first-Bibliothek für **Python-basierte CAD-Modellierung**
- Implementiert als Plugin für FreeCAD



CadQuery 2

Python-Bibliothek für parametrisches CAD (2020)

- Neuimplementierung von CadQuery als direkter Wrapper um **OCCT** (ohne

FreeCAD-Abhängigkeit)
- Erweiterte B-Rep-Funktionalität
- Integration mit Jupyter Notebooks

Build123d

Python-Bibliothek für parametrisches CAD (2022)

- Entwickelt als **Alternative zu CadQuery 2** mit veränderter Syntax
- Basiert ebenfalls auf **OCCT**
- Geometrie-Objekte können mit CadQuery-Code ausgetauscht werden

Fazit: moderner Python-CAD-Stack

- **OCCT** als Geometrie-Kernel
- **CadQuery 2** oder **Build123d** als Python-Bibliotheken für parametrische Modellierung

Visualisierungs-Tools:

- **Jupyter-CadQuery** für interaktive Visualisierung in Jupyter Notebooks
- **OCP CAD Viewer for VS Code** für interaktive Visualisierung in Visual Studio Code

Demo: unsere erste CAD-Geometrie in Python (build123d-Version)

```
import jupyter_cadquery
import build123d as bd

length = 80
```

```
width = 60
height = 10

base = bd.Box(length, width, height)
hole = bd.Cylinder(radius=width / 4, height=height)
part = base - hole

top_f = part.faces().sort_by(bd.Axis.Z).last
hole_edges = top_f.edges().filter_by(bd.GeomType.CIRCLE)
result = part.fillet(radius=2, edge_list=hole_edges)

result
```

Demo: unsere erste CAD-Geometrie in Python (CadQuery-Version)

```
import jupyter_cadquery
from cadquery import func as cf

length = 80
width = 60
height = 10

base = cf.box(length, width, height)
hole = cf.cylinder(d=width / 2, h=height)
part = base - hole

top_f = part.faces(">Z")
hole_edge = top_f.edges("%CIRCLE")
result = part.fillet(2.0, [hole_edge])

result
```

Zusammenfassung

- **Parametrische Modellierung mit Code**: Versionsverwaltung, Automatisierung, Optimierung

- **Python als CAD-Sprache**: Allzweck, plattformunabhängig, weit verbreitet
- **Drei Geometrie-Darstellungen**: CSG, B-Rep, Mesh
  - CSG: keine Freiformflächen, nicht allgemein genug
  - B-Rep: Standard in professionellen CAD-Systemen → von uns verwendet
  - Mesh: für 3D-Druck (CAM) und Finite-Elemente-Simulation (CAE) relevant, nicht für Konstruktion
- **OCCT**: Open-Source-Kernel mit B-Rep
- **CadQuery 2 und Build123d**: Moderne Python-Bibliotheken für parametrisches CAD