# KURS OPENSCAD

Modelowanie parametryczne CAD dla programistów

---

## CZĘŚĆ 1

Podstawy – Bryły, Transformacje, Operacje boolowskie

Materiały dydaktyczne | INF.02 / INF.03

Opracowanie: mgr Marek Spurek

# Spis treści

# 1. Wprowadzenie do OpenSCAD

OpenSCAD to darmowe, otwartoźródłowe oprogramowanie do modelowania 3D. W odróżnieniu od klasycznych programów CAD (Fusion 360, FreeCAD) nie korzystamy tu z myszy do rysowania brył – zamiast tego piszemy kod w języku skryptowym podobnym do C. Efektem jest model 3D gotowy do druku lub obróbki CNC.

**Dlaczego OpenSCAD?**

- Modele są w pełni parametryczne – zmiana jednej zmiennej zmienia cały projekt.
- Pliki .scad to zwykły tekst – można nimi zarządzać w Git.
- Doskonały do druku 3D (eksport STL, AMF, 3MF).
- Bezpłatny, działa na Windows / macOS / Linux.
- Świetny punkt wejścia do myślenia algorytmicznego w CAD.

> **OpenSCAD a programowanie**
> Jeśli znasz już podstawy programowania (zmienne, pętle, funkcje), nauka OpenSCAD zajmie ci bardzo mało czasu. To tak naprawdę język programowania, którego wynikiem jest obiekt 3D, a nie tekst na ekranie.

**Zasada działania:**

Piszesz kod → klikasz F5 (podgląd) lub F6 (renderowanie) → widzisz bryłę 3D → eksportujesz do STL → drukujesz.

# 2. Instalacja i interfejs

## 2.1 Instalacja

Pobierz OpenSCAD ze strony: https://openscad.org/downloads.html

**Dostępne wersje:**

- Windows: instalator .exe lub wersja portable
- macOS: plik .dmg
- Linux: pakiet .AppImage lub z repozytorium dystrybucji

> ⚠ **Wersja stabilna vs nightly**
> Zalecaj uczniom pobranie wersji stabilnej (np. 2021.01 lub nowszej). Wersja nightly zawiera nowości, ale może mieć błędy.

## 2.2 Interfejs programu

| | |
|---|---|
| Panel edytora (lewy) | Tu piszesz kod OpenSCAD |
| Okno podglądu 3D (prawy) | Wizualizacja modelu w czasie rzeczywistym |
| Konsola (dół) | Błędy, ostrzeżenia, echo() – wyniki debugowania |
| F5 | Podgląd (szybki, przybliżony) |
| F6 | Pełne renderowanie (wolniejsze, dokładne) |
| F7 | Eksport do STL/AMF/DXF |

## 2.3 Twój pierwszy skrypt

Wpisz poniższy kod w edytorze i naciśnij F5:

```
OpenSCAD
// Mój pierwszy model w OpenSCAD
// Komentarze zaczynają się od //

cube([20, 30, 10]);   // sześcian 20 x 30 x 10 mm
```

Powinieneś zobaczyć szarą prostopadłościenną bryłę. Gratulacje – to twój pierwszy model 3D!

# 3. Podstawowe bryły 3D

OpenSCAD posiada kilka wbudowanych brył (prymitywów). Wszystkie inne modele budujesz przez ich łączenie i modyfikowanie.

## 3.1 Sześcian – cube()

**Składnia:**

```
OpenSCAD
cube(rozmiar);                    // sześcian o boku = rozmiar
cube([szerokość, głębokość, wysokość]);  // prostopadłościan
cube([x, y, z], center = true);   // wyśrodkowany względem punktu 0
```

Przykłady:

```
OpenSCAD
// Przykład 1a – zwykły sześcian 10 mm
cube(10);

// Przykład 1b – prostopadłościan 40 x 25 x 15 mm
cube([40, 25, 15]);

// Przykład 1c – sześcian wyśrodkowany
// (środek bryły = punkt [0,0,0])
cube([30, 30, 30], center = true);

// Przykład 1d – płaska prostokątna płyta
cube([100, 60, 2]);
```

> **Układ współrzędnych**
> Domyślnie cube() ma narożnik w punkcie [0,0,0]. Z parametrem center=true środek bryły leży w [0,0,0]. Jednostki są milimetrami – OpenSCAD nie ma własnej jednostki, interpretujesz ją sam.

## 3.2 Sfera – sphere()

**Składnia:**

```
OpenSCAD
sphere(r = promień);          // przez promień
sphere(d = średnica);         // przez średnicę
sphere(r = 10, $fn = 50);     // $fn = liczba segmentów (gładkość)
```

Przykłady:

```
OpenSCAD
// Przykład 2a – sfera o promieniu 15 mm
sphere(r = 15);

// Przykład 2b – sfera o średnicy 30 mm
sphere(d = 30);

// Przykład 2c – gładka sfera (więcej segmentów)
sphere(r = 20, $fn = 100);

// Przykład 2d – sfera mała, niska rozdzielczość (szybszy podgląd)
```

```
sphere(r = 5, $fn = 12);

// Przykład 2e – użycie $fa i $fs zamiast $fn
// $fa = minimalny kąt segmentu (stopnie)
// $fs = minimalna długość segmentu (mm)
sphere(r = 15, $fa = 5, $fs = 0.5);
```

**$fn – najważniejszy parametr jakości**

$fn określa liczbę segmentów bryły obrotowej. Przy $fn=12 sfera wygląda jak graniastosłup, przy $fn=100 jest gładka. W podglądzie używaj małych wartości ($fn=20), do finalnego STL – dużych ($fn=100 lub więcej).

## 3.3 Walec i stożek – cylinder()

**Składnia:**

OpenSCAD
```
cylinder(h, r, center=false);          // walec
cylinder(h, r1, r2, center=false);     // stożek/ścięty stożek
cylinder(h=wys, d=śred, $fn=50);       // parametry nazwane
```

Przykłady:

OpenSCAD
```
// Przykład 3a – walec o promieniu 10, wysokości 20
cylinder(h=20, r=10);

// Przykład 3b – walec przez średnicę
cylinder(h=30, d=25);

// Przykład 3c – walec wyśrodkowany pionowo
cylinder(h=40, r=8, center=true);

// Przykład 3d – stożek (r1=dół, r2=góra)
cylinder(h=30, r1=15, r2=0);

// Przykład 3e – ścięty stożek (frustum)
cylinder(h=25, r1=20, r2=10);

// Przykład 3f – rurka (walec minus walec wewnętrzny)
difference() {
  cylinder(h=40, r=15, $fn=60);      // zewnętrzny
  cylinder(h=42, r=12, $fn=60);      // wewnętrzny (nieco wyższy!)
}

// Przykład 3g – sześciokąt (pryzmat 6-ścienny)
cylinder(h=10, r=15, $fn=6);
```

**Rurka – dlaczego walec wewnętrzny jest wyższy?**

Przy operacji difference() oba walce muszą się przeciąć na całej długości. Gdyby oba miały h=40, ściany mogą mieć grubość 0 i OpenSCAD generuje błędy geometryczne. Zawsze rób odejmowany element o 1-2 mm dłuższy z obu stron.

## 3.4 Wielościan – polyhedron()

polyhedron() pozwala tworzyć dowolne bryły przez podanie listy wierzchołków i ścian. Przydatny do zaawansowanych kształtów, które nie dadzą się złożyć z prymitywów.

OpenSCAD

```
// Przykład 4a – czworościan (tetrahedron)
polyhedron(
  points = [
    [0,  0,  0],   // 0 – podstawa przód-lewo
    [10, 0,  0],   // 1 – podstawa przód-prawo
    [5,  10, 0],   // 2 – podstawa tył
    [5,  5,  10],  // 3 – wierzchołek
  ],
  faces = [
    [0, 1, 2],   // podstawa
    [0, 3, 1],   // ściana przód
    [1, 3, 2],   // ściana prawo
    [2, 3, 0],   // ściana lewo
  ]
);

// Przykład 4b – piramida czworoboczna 20x20, wys. 15
polyhedron(
  points = [
    [0,0,0],[20,0,0],[20,20,0],[0,20,0],  // podstawa
    [10,10,15]                             // wierzchołek
  ],
  faces = [
    [0,3,2,1],  // podstawa (odwrotna kolejność = skierowana w dół)
    [0,1,4],    // ściany boczne
    [1,2,4],
    [2,3,4],
    [3,0,4]
  ]
);
```

> ⚠ **Kierunek normalnych w polyhedron()**
> Kolejność wierzchołków na ścianie musi być zgodna z regułą prawej ręki: patrząc na ścianę z zewnątrz, wierzchołki idą przeciwnie do ruchu wskazówek zegara. Błędna kolejność tworzy 'odwróconą' bryłę – OpenSCAD poinformuje o tym w konsoli.

# 4. Transformacje

Transformacje przesuwają, obracają i skalują bryły. W OpenSCAD działają jak funkcje opakowujące – umieszczasz bryłę wewnątrz nawiasów transformacji.

## 4.1 Przesunięcie – translate()

OpenSCAD

```
translate([dx, dy, dz]) obiekt;

// Przykład 5a – trzy sześciany obok siebie (oś X)
translate([0,  0, 0]) cube([10,10,10]);
translate([15, 0, 0]) cube([10,10,10]);
translate([30, 0, 0]) cube([10,10,10]);

// Przykład 5b – dwie sfery na osi Z
translate([0, 0,  0])  sphere(r=8, $fn=40);
translate([0, 0, 20])  sphere(r=8, $fn=40);

// Przykład 5c – walec umieszczony centralnie nad sześcianem
cube([20, 20, 10]);
translate([10, 10, 10])  // przesuwamy do środka górnej ściany
  cylinder(h=15, r=5, $fn=40);

// Przykład 5d – siatka 4x3 sześcianów
for (x = [0:3]) {
  for (y = [0:2]) {
    translate([x*15, y*15, 0]) cube([12,12,12]);
  }
}
```

## 4.2 Obrót – rotate()

OpenSCAD

```
rotate([rx, ry, rz]) obiekt;   // kąty w stopniach, kolejność: X→Y→Z
rotate(a, v) obiekt;           // obrót o kąt a wokół wektora v

// Przykład 6a – walec leżący (obrót 90° wokół osi Y)
rotate([0, 90, 0])
  cylinder(h=30, r=5, $fn=40);

// Przykład 6b – obrót kostki o 45° wokół osi Z
rotate([0, 0, 45])
  cube([20, 20, 10]);

// Przykład 6c – łopata wentylatora (obrót względem osi arbitralnej)
rotate(a=45, v=[1, 1, 0])
  cube([5, 30, 2]);

// Przykład 6d – 6 walców rozmieszczonych po okręgu (360/6 = 60°)
for (i = [0:5]) {
  rotate([0, 0, i*60])
    translate([20, 0, 0])
      cylinder(h=10, r=3, $fn=30);
}

// Przykład 6e – krzyż z 4 walców
for (i = [0:3]) {
  rotate([0, 0, i*90])
```

```
    translate([15, 0, 5])
      cylinder(h=10, r=4, center=true, $fn=30);
}
cylinder(h=10, r=6, $fn=30);  // środkowy walec
```

## 4.3 Skalowanie – scale() i resize()

OpenSCAD

```
scale([sx, sy, sz]) obiekt;   // współczynniki (1.0 = bez zmiany)
resize([nx, ny, nz]) obiekt;  // nowe wymiary w mm

// Przykład 7a – elipsoida (sfera spłaszczona)
scale([1.5, 1, 0.5])  sphere(r=10, $fn=60);

// Przykład 7b – podwójne powiększenie na osi Z
scale([1, 1, 2])  cube([10, 10, 10]);

// Przykład 7c – zmiana rozmiaru walca do 20x20x30
resize([20, 20, 30])  cylinder(h=10, r=5, $fn=40);

// Przykład 7d – skalowanie proporcjonalne przez resize (0 = auto)
resize([30, 0, 0], auto=true)  sphere(r=5, $fn=40);
// sfera o promieniu 5 (śred=10) zostanie przeskalowana do śred=30
// pozostałe osie skalowane proporcjonalnie
```

## 4.4 Symetria – mirror()

OpenSCAD

```
mirror([nx, ny, nz]) obiekt;  // odbicie względem płaszczyzny normalnej
[n]

// Przykład 8a – odbicie względem płaszczyzny YZ (oś X = 0)
cube([20, 10, 5]);
mirror([1, 0, 0])  cube([20, 10, 5]);

// Przykład 8b – klips symetryczny
module polowa_klipsa() {
  difference() {
    cube([15, 5, 20]);
    translate([3, -1, 3]) cube([9, 7, 14]);
  }
}
polowa_klipsa();
mirror([1, 0, 0]) polowa_klipsa();

// Przykład 8c – czwórkowe odbicie (X i Y)
module cwiartka() {
  translate([5, 5, 0]) cylinder(h=10, r=3, $fn=30);
}
cwiartka();
mirror([1,0,0]) cwiartka();
mirror([0,1,0]) cwiartka();
mirror([1,0,0]) mirror([0,1,0]) cwiartka();
```

# 5. Operacje boolowskie

Operacje boolowskie to serce OpenSCAD. Łączysz bryły przez sumę, odejmujesz od siebie lub wycinasz część wspólną. Każda operacja opakowuje dwie lub więcej brył.

## 5.1 Suma – union()

union() łączy bryły w jedną (domyślna operacja – dwie bryły obok siebie bez żadnej operacji też tworzą unię).

OpenSCAD

```
// Przykład 9a - proste połączenie
union() {
  cube([30, 10, 10]);
  translate([10, 10, 0]) cube([10, 20, 10]);
}

// Przykład 9b - kształt litery 'T'
union() {
  cube([40, 10, 5]);           // pozioma belka
  translate([15, 10, 0])       // pionowa belka
    cube([10, 30, 5]);
}

// Przykład 9c - kula na walcu (zabawkowy kształt)
union() {
  cylinder(h=20, r=5, $fn=40);
  translate([0, 0, 20])
    sphere(r=8, $fn=40);
}
```

## 5.2 Różnica – difference()

difference() odejmuje drugą i kolejne bryły od pierwszej. Pierwsza bryła to 'materiał', pozostałe to 'otwory/kieszenie'.

OpenSCAD

```
// Przykład 10a - sześcian z otworem przelotowym
difference() {
  cube([30, 30, 20]);
  translate([15, 15, -1])       // -1: wychodzimy poza dolną ścianę
    cylinder(h=22, r=6, $fn=40);
}

// Przykład 10b - pudełko (sześcian z kieszenią)
difference() {
  cube([40, 30, 20]);           // zewnętrzne
  translate([2, 2, 2])          // wewnętrzne (ściany 2 mm)
    cube([36, 26, 19]);         // 19: zostawiamy dno
}

// Przykład 10c - krążek z dziurą (washer)
difference() {
  cylinder(h=3, r=15, $fn=60); // zewnętrzny
  translate([0, 0, -1])
    cylinder(h=5, r=6, $fn=60); // otwór
}

// Przykład 10d - cegła z 3 otworami
difference() {
```

```
  cube([60, 25, 20]);
  for (x = [12, 30, 48]) {
    translate([x, -1, 10])
      rotate([-90, 0, 0])
        cylinder(h=27, r=5, $fn=30);
  }
}

// Przykład 10e – gwintowany kołek (przybliżony)
difference() {
  cylinder(h=30, r=6, $fn=6);   // sześciokąt = łeb
  translate([0, 0, -1])
    cylinder(h=32, r=2.5, $fn=20); // otwór na gwint
}
```

## 5.3 Część wspólna – intersection()

intersection() zostawia tylko tę część przestrzeni, którą zajmują WSZYSTKIE bryły jednocześnie.

OpenSCAD

```
// Przykład 11a – przecięcie sześcianu i sfery = zaokrąglona kostka
intersection() {
  cube([20, 20, 20], center=true);
  sphere(r=13, $fn=60);
}

// Przykład 11b – cylinder przechodzący przez sześcian
intersection() {
  cube([30, 30, 30], center=true);
  cylinder(h=60, r=12, center=true, $fn=50);
}

// Przykład 11c – walec i walec prostopadle (kształt beczki)
intersection() {
  cylinder(h=40, r=15, center=true, $fn=60);
  rotate([90,0,0])
    cylinder(h=40, r=15, center=true, $fn=60);
}

// Przykład 11d – logo soczewkowe
intersection() {
  translate([-5, 0, 0]) cylinder(h=3, r=15, $fn=50);
  translate([ 5, 0, 0]) cylinder(h=3, r=15, $fn=50);
}
```

# 6. Zmienne i parametry

OpenSCAD używa zmiennych statycznych – raz przypisanej wartości NIE MOŻNA zmienić w trakcie wykonania. To czyni model w pełni deterministycznym i parametrycznym.

**OpenSCAD**

```
// Przykład 12a – parametryczny prostopadłościan
szerokosc = 40;   // [mm]
glebokosc = 30;
wysokosc  = 15;
grubosc_sciany = 2;

difference() {
  cube([szerokosc, glebokosc, wysokosc]);
  translate([grubosc_sciany, grubosc_sciany, grubosc_sciany])
    cube([szerokosc - 2*grubosc_sciany,
          glebokosc - 2*grubosc_sciany,
          wysokosc]);   // wyższy: brak dna na górze
}

// Przykład 12b – parametryczny śrubokręt do druku 3D
dlugosc_trzonka = 80;     // mm
srednica_trzonka = 8;     // mm
bok_glowicy = 20;         // mm
wysokosc_glowicy = 25;    // mm

// trzonek
cylinder(h=dlugosc_trzonka, d=srednica_trzonka, $fn=30);
// głowica – sześciokąt
translate([0, 0, dlugosc_trzonka])
  cylinder(h=wysokosc_glowicy, d=bok_glowicy, $fn=6);

// Przykład 12c – customizer (widoczny w OpenSCAD Customizer)
/* [Wymiary skrzynki] */
dlugosc = 80;    // [10:200]
szerokosc2 = 50;  // [10:200]
wysokosc2 = 40;   // [10:200]
grubosc2 = 2;     // [1:5]

difference() {
  cube([dlugosc, szerokosc2, wysokosc2]);
  translate([grubosc2, grubosc2, grubosc2])
    cube([dlugosc-2*grubosc2, szerokosc2-2*grubosc2, wysokosc2]);
}
```

**Customizer w OpenSCAD**

Komentarze /* [Sekcja] */ i wartości // [min:max] aktywują Customizer – panel z suwakami, który pozwala zmieniać parametry bez edycji kodu. Świetne narzędzie dla uczniów!

# 7. Pętle i moduły

## 7.1 Pętla for

OpenSCAD

```
// Pętla po zakresie
for (i = [0:4]) {          // i = 0, 1, 2, 3, 4
  translate([i*15, 0, 0]) sphere(r=5, $fn=30);
}

// Pętla ze skokiem
for (i = [0:10:60]) {      // i = 0, 10, 20, 30, 40, 50, 60
  translate([i, 0, 0]) cube([8,8,8]);
}

// Pętla po liście wartości
for (r = [3, 5, 8, 12]) {
  translate([r*2.5, 0, 0]) sphere(r=r, $fn=30);
}

// Pętla 2D → siatka
for (x=[0:3], y=[0:3]) {
  translate([x*12, y*12, 0]) cube([10,10,10]);
}
```

## 7.2 Instrukcja warunkowa if

OpenSCAD

```
// Przykład 14 - różne kształty w zależności od zmiennej
ksztalt = "walec";  // "walec", "sfera", "szescian"

if (ksztalt == "walec") {
  cylinder(h=20, r=8, $fn=40);
} else if (ksztalt == "sfera") {
  sphere(r=10, $fn=40);
} else {
  cube([15, 15, 15]);
}
```

## 7.3 Moduły – własne funkcje

OpenSCAD

```
// Definicja modułu
module nazwa_modulu(parametr1, parametr2 = domyslna_wartosc) {
  // ciało modułu
}

// Przykład 15a - moduł: zaokrąglona belka
module belka(dl, sz, wys, r_zaokr = 3) {
  hull() {
    translate([r_zaokr, r_zaokr, 0]) cylinder(h=wys, r=r_zaokr,
$fn=30);
    translate([dl-r_zaokr, r_zaokr, 0]) cylinder(h=wys, r=r_zaokr,
$fn=30);
    translate([r_zaokr, sz-r_zaokr, 0]) cylinder(h=wys, r=r_zaokr,
$fn=30);
    translate([dl-r_zaokr, sz-r_zaokr, 0]) cylinder(h=wys, r=r_zaokr,
$fn=30);
```

```
  }
}

belka(60, 30, 8);                    // domyślne zaokrąglenie 3 mm
translate([0, 40, 0]) belka(60, 30, 8, r_zaokr=8);   // duże
zaokrąglenie

// Przykład 15b – moduł: śruba (uproszczona)
module sruba(dl_trzonka, d_trzonka, d_lba, wys_lba) {
  cylinder(h=dl_trzonka, d=d_trzonka, $fn=20);
  translate([0, 0, dl_trzonka])
    cylinder(h=wys_lba, d=d_lba, $fn=6);
}

sruba(20, 4, 8, 5);
translate([15, 0, 0]) sruba(30, 5, 10, 6);

// Przykład 15c – moduł rekurencyjny (drzewo)
module galaz(dlugosc, poziom) {
  if (poziom > 0) {
    cylinder(h=dlugosc, r=poziom*0.8, $fn=8);
    translate([0, 0, dlugosc]) {
      rotate([0, 30, 0])  galaz(dlugosc*0.7, poziom-1);
      rotate([0, -30, 60]) galaz(dlugosc*0.7, poziom-1);
    }
  }
}

galaz(20, 4);   // drzewo 4 poziomów
```

# 8. Projekty praktyczne

Poniżej znajdziesz 10 kompletnych projektów w rosnącym stopniu trudności. Każdy projekt to gotowy do uruchomienia kod OpenSCAD.

## Projekt 1: Podstawka pod telefon

Prosta podstawka drukowana w jednym kawałku, z regulowanym kątem.

OpenSCAD

```
// PROJEKT 1 – Podstawka pod telefon
kat = 70;          // kąt pochylenia [45-80]
szerokosc = 80;    // szerokość telefonu + luz [mm]
glebokosc = 25;    // głębokość podstawy [mm]
grubosc = 3;       // grubość ścianki [mm]

// Podstawa pozioma
cube([szerokosc, glebokosc, grubosc]);

// Tylna ścianka pochylona
translate([0, glebokosc, 0])
  rotate([-kat+90, 0, 0])
    cube([szerokosc, 60, grubosc]);

// Przedni zderzak (zapobiega ześlizgiwaniu)
cube([szerokosc, grubosc, 8]);
```

## Projekt 2: Uchwyt na kable – klips

Klips zaciskający się na krawędzi biurka lub półki.

OpenSCAD

```
// PROJEKT 2 – Klips na kable
szerokosc_klipsa = 20;
grubosc_klipsa = 3;
srednica_otworu = 8;  // średnica kabla
glebokosz_uchwytu = 15; // grubość blatu

difference() {
  union() {
    // Górna gałąź
    cube([szerokosc_klipsa, 40, grubosc_klipsa]);
    // Pionowa część
    translate([0, 37, 0])
      cube([szerokosc_klipsa, grubosc_klipsa, glebokosz_uchwytu+10]);
    // Dolna gałąź
    translate([0, 37, glebokosz_uchwytu+10-grubosc_klipsa])
      cube([szerokosc_klipsa, 20, grubosc_klipsa]);
  }
  // Otwór na kabel
  translate([szerokosc_klipsa/2, 15, -1])
    cylinder(h=grubosc_klipsa+2, d=srednica_otworu, $fn=30);
}
```

## Projekt 3: Kostka do gry (parametryczna)

Sześcian z oczkami wybity jako wcięcia – w pełni parametryczny.

OpenSCAD

```
// PROJEKT 3 - Kostka do gry
a = 20;          // bok kostki [mm]
r_kant = 2;      // promień zaokrąglenia krawędzi
r_oczko = 2.2;   // promień oczka
gl_oczka = 1;    // głębokość wgniecenia

// Zaokrąglona kostka - hull() z kul
module zaokr_kostka(b, r) {
  hull()
    for (x=[r, b-r], y=[r, b-r], z=[r, b-r])
      translate([x,y,z]) sphere(r=r, $fn=20);
}

module oczko(x, y, z) {
  translate([x, y, z])
    sphere(r=r_oczko, $fn=20);
}

difference() {
  zaokr_kostka(a, r_kant);
  // Ściana 1 (góra) - jedno oczko
  oczko(a/2, a/2, a-gl_oczka);
  // Ściana 6 (dół) - 6 oczek
  for (x=[a*0.25, a*0.75], y=[a*0.2, a*0.5, a*0.8])
    oczko(x, y, gl_oczka);
  // Ściana 2 (przód) - dwa oczka
  for (p=[[a*0.3, a-gl_oczka, a*0.7],[a*0.7, a-gl_oczka, a*0.3]])
    oczko(p[0], p[1], p[2]);
  // (pozostałe ściany pominięte dla zwięzłości)
}
```

## Projekt 4: Mini skrzynka z pokrywką

Pudełko z pasującą pokrywą – użyj tolerancji 0.2 mm dla wydruku.

OpenSCAD

```
// PROJEKT 4 - Skrzynka z pokrywką
dl = 60; sz = 40; wys = 30; gr = 2; tolerancja = 0.3;

module pudelko() {
  difference() {
    cube([dl, sz, wys]);
    translate([gr, gr, gr])
      cube([dl-2*gr, sz-2*gr, wys]);
  }
}

module pokrywka() {
  wys_p = 8;
  difference() {
    cube([dl, sz, wys_p]);
    translate([gr, gr, -1])
      cube([dl-2*gr, sz-2*gr, wys_p]);
    // wpust wewnętrzny
    translate([gr+tolerancja, gr+tolerancja, gr])
      cube([dl-2*gr-2*tolerancja, sz-2*gr-2*tolerancja, wys_p]);
  }
}

pudelko();
```

```
translate([0, sz+10, 0]) pokrywka();
```

## Projekt 5: Koło zębate – uproszczone

Koło zębate z parametryczną liczbą i kształtem zębów.

OpenSCAD

```
// PROJEKT 5 - Koło zębate (uproszczone)
n_zebow = 16;       // liczba zębów
modul = 2;          // moduł (mm/ząb)
grubosc = 8;        // grubość koła
r_osi = 3;          // promień otworu osi

r_pitch  = n_zebow * modul / 2;   // promień podziałowy
r_glowa  = r_pitch + modul;       // promień głowy zęba
r_stop   = r_pitch - 1.25*modul; // promień stopy

difference() {
  union() {
    cylinder(h=grubosc, r=r_stop, $fn=60);
    for (i=[0:n_zebow-1]) {
      rotate([0, 0, i*(360/n_zebow)])
        translate([r_pitch, 0, 0])
          hull() {
            cylinder(h=grubosc, r=modul*0.6, $fn=12);
            translate([modul, 0, 0])
              cylinder(h=grubosc, r=modul*0.3, $fn=12);
          }
    }
  }
  // otwór osi
  translate([0, 0, -1]) cylinder(h=grubosc+2, r=r_osi, $fn=30);
}
```

## Projekt 6: Uchwyt na szklankę – pierścień z nóżkami

Podstawek / uchwyt termiczny na kubek.

OpenSCAD

```
// PROJEKT 6 - Podstawek na kubek
d_kubka = 80;       // średnica zewnętrzna kubka [mm]
wys_uch = 15;       // wysokość uchwytu
gr_pier = 3;        // grubość pierścienia
n_nozek = 4;        // liczba nóżek
wys_noz = 5;        // wysokość nóżek
r_noz   = 5;        // promień nóżki

// Pierścień
difference() {
  cylinder(h=wys_uch, d=d_kubka+2*gr_pier, $fn=80);
  translate([0,0,-1])
    cylinder(h=wys_uch+2, d=d_kubka, $fn=80);
}

// Nóżki
for (i=[0:n_nozek-1]) {
  rotate([0, 0, i*(360/n_nozek)])
    translate([(d_kubka/2+gr_pier/2), 0, -wys_noz])
      cylinder(h=wys_noz, r=r_noz, $fn=20);
```

```
}
```

## Projekt 7: Kluczyk do kół – adapter

Hexagonalny adapter 1/2-cala na klucz 17 mm.

```
OpenSCAD
// PROJEKT 7 - Adapter do klucza nasadkowego
d_zewn = 30;        // średnica zewnętrzna
wys    = 25;        // wysokość adaptera
klucz_wew = 17;   // rozmiar sześciokąta wewnętrznego (SW17)
klucz_zew = 12;   // rozmiar sześciokąta zewnętrznego

difference() {
  // korpus
  cylinder(h=wys, d=d_zewn, $fn=6);
  // wnęka na klucz wewnętrzny (SW17)
  translate([0, 0, -1])
    cylinder(h=14, d=klucz_wew/cos(30), $fn=6);
  // czop zewnętrzny (SW12) - odejmujemy od góry
  translate([0, 0, 12])
    difference() {
      cylinder(h=wys, d=d_zewn+2, $fn=6);
      cylinder(h=wys, d=klucz_zew/cos(30), $fn=6);
    }
}
```

## Projekt 8: Labirynt – generator

Prosta ścianka labiryntu generowana pętlą for i tablicą danych.

```
OpenSCAD
// PROJEKT 8 - Prosta ścianka labiryntu
// Dane labiryntu: 1 = ściana, 0 = przejście
mapa = [
  [1,1,1,1,1,1,1],
  [1,0,0,0,1,0,1],
  [1,0,1,0,0,0,1],
  [1,0,1,1,1,0,1],
  [1,0,0,0,0,0,1],
  [1,1,1,1,1,1,1],
];

komorka = 10;  // rozmiar komórki mm
wys_sciany = 8; // wysokość ścian mm

for (wiersz = [0:len(mapa)-1]) {
  for (kol = [0:len(mapa[wiersz])-1]) {
    if (mapa[wiersz][kol] == 1) {
      translate([kol*komorka, wiersz*komorka, 0])
        cube([komorka, komorka, wys_sciany]);
    }
  }
}
```

## Projekt 9: Gwiazdka dekoracyjna

Wieloramienna gwiazdka z parametryczną liczbą ramion.

OpenSCAD

```
// PROJEKT 9 - Gwiazdka dekoracyjna
n_ramion = 6;      // liczba ramion
r_wew    = 8;      // promień wewnętrzny
r_zew    = 20;     // promień zewnętrzny
grubosc = 3;       // grubość gwiazdki
r_otw    = 2;      // otwór do zawieszenia

katy_wew = [for (i=[0:n_ramion-1]) i*(360/n_ramion) + 90/n_ramion*0];

module gwiazdka() {
  linear_extrude(height=grubosc) {
    polygon([
      for (i=[0:n_ramion-1]) each [
        [r_zew*cos(i*360/n_ramion+90),
         r_zew*sin(i*360/n_ramion+90)],
        [r_wew*cos((i+0.5)*360/n_ramion+90),
         r_wew*sin((i+0.5)*360/n_ramion+90)]
      ]
    ]);
  }
}

difference() {
  gwiazdka();
  translate([0, r_zew-3, grubosc/2])
    rotate([90,0,0])
      cylinder(h=4, r=r_otw, $fn=20, center=true);
}
```

## Projekt 10: Imię w 3D – linear_extrude + text()

Wypukły napis gotowy do druku.

OpenSCAD

```
// PROJEKT 10 - Napis 3D
napis = "OPENSCAD";
wysokosc_napisu = 8;  // rozmiar czcionki [mm]
wysuniecie = 5;       // grubość liter [mm]
podstawa = true;      // czy dodać płytę pod napis

module tekst_3d(str, h_font, h_extr) {
  linear_extrude(height=h_extr) {
    text(str, size=h_font, font="Liberation Sans:style=Bold",
         halign="center", valign="center");
  }
}

tekst_3d(napis, wysokosc_napisu, wysuniecie);

// Opcjonalna podstawa
if (podstawa) {
  translate([0, 0, -2])
    cube([len(napis)*wysokosc_napisu*0.65 + 10,
          wysokosc_napisu + 8, 2],
         center=true);
}
```

# 9. Zadania do samodzielnego wykonania

Poniższe zadania mają różny poziom trudności. Poziom 1★ – podstawowy, 3★★★ – zaawansowany.

| | | | |
|---|---|---|---|
| 1 | Stwórz kulę na walcu (bałwan) – 3 sfery i 2 walce | ★☆☆ | sphere, cylinder, translate |
| 2 | Zaprojektuj plastikową kostkę Rubika (bez mechanizmu) – siatka 3x3x3 małych sześcianów ze szczelinami | ★☆☆ | cube, for, translate |
| 3 | Utwórz uchyt do pióra/długopisu – pierścień z płytką montażową | ★★☆ | cylinder, difference, mirror |
| 4 | Zaprojektuj klucz meblowy imbusowy (L-shape) z sześciokątnym końcem | ★★☆ | cylinder($fn=6), rotate, union |
| 5 | Stwórz parametryczną tabliczkę z imieniem do druku 3D | ★★☆ | text, linear_extrude, difference |
| 6 | Zaprojektuj koszyczek na jajko wielkanocne – walec z ażurową siatką | ★★★ | cylinder, for, intersection/difference |
| 7 | Stwórz kompletny model domku – ściany, dach, drzwi, okna – wszystko parametrycznie | ★★★ | moduły, cube, difference, hull |

**Wskazówka dla nauczyciela**

Zadania 1-3 nadają się na pierwsze zajęcia, 4-5 na drugie. Zadania 6-7 mogą być projektem zaliczeniowym. Poproś uczniów o eksport STL i wydruk modelu na szkolnej drukarce 3D – to świetna motywacja!

# Podsumowanie – Część 1

W tej części poznałeś:

- Cztery podstawowe bryły: cube, sphere, cylinder, polyhedron
- Cztery transformacje: translate, rotate, scale/resize, mirror
- Trzy operacje boolowskie: union, difference, intersection
- Zmienne i parametryzację modeli
- Pętle for, instrukcje warunkowe if
- Moduły – własne, wielokrotnego użytku bloki kodu
- 10 projektów praktycznych gotowych do druku

**W Części 2 nauczymy się:**

- Kształtów 2D i ich ekstruzji (linear_extrude, rotate_extrude)
- Operacji hull() i minkowski()
- Importu plików SVG i DXF
- Zaawansowanych modułów i rekurencji
- Optymalizacji modeli pod druk 3D (ściany, mosty, podparcia)

**Zasoby**

Dokumentacja: https://openscad.org/documentation.html | Galeria modeli: https://www.printables.com | MCAD – biblioteka modułów: https://github.com/openscad/MCAD