INFORMATIKA

3D modelování v OpenSCAD

MARKÉTA TRNEČKOVÁ

Přírodovědecká fakulta UP, Olomouc

Zejména kvůli masivnímu rozšíření technologie 3D tisku se přímo nabízí zařazení základů 3D modelování do výuky. Žáci středních škol, zejména techničtěji zaměřených, ale i gymnázií, mohou s poměrně skromnými znalostmi provádět vlastní návrh rozličných komponent a tím lépe porozumět technickým principům konstrukce. Parametrické 3D modelování navíc umožňuje žákům seznámit se s jednoduchými základy programování. Na základních školách může pomoci s rozvojem prostorové představivosti nebo sloužit k demonstraci různých matematických konceptů, jako jsou např. geometrické tvary, geometrické transformace a množinové operace.

3D modelování

Ve světě 3D modelování existuje bohatý výběr softwarových nástrojů, které slouží k vytváření trojrozměrných objektů. Ty se liší způsoby, jakými je model vytvářen. Objekty je možné kreslit (pomocí myši) v různých *CAD software*[1)] jako jsou například TinkerCAD, FreeCAD nebo OpenSCAD[2)], kterému se budeme věnovat. CAD je založený na konstruktivní geometrii, kde je objekt reprezentován popisem konstrukce z jednoduchých těles, kterým říkáme geometrická primitiva[3)] [2]. Modelům, které je možné popsat pomocí geometrických primitiv, se také říká *hard solid*. V CAD však není možné vymodelovat objekty, které takto popsat nejde, například obličej nebo postavu. Takové objekty je potřeba modelovat pomocí modelovacích

[1)] CAD = Computer Aided Design, česky počítačem podporované projektování.
[2)] Čteme jako Open-S-CAD.
[3)] Geometrickými primitivy mohou být například koule, krychle, ale i různé plochy.

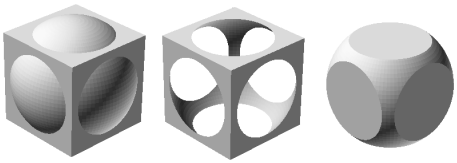
programů určených pro umělecké modelovaní, jakým je například Blender[4]. Zde je objekt reprezentován jako síť bodů a ploch (mesh), která se pomocí různých nástrojů upravuje.

OpenSCAD se od ostatních CAD výrazně odlišuje. V OpenSCAD se modeluje pomocí jednoduchého programovacího jazyka (jenž se nazývá také OpenSCAD), kterým popisujeme postup konstrukce výsledného objektu. Díky tomu má designér větší kontrolu nad modelovaným objektem a umožňuje mu to kdykoliv v průběhu jednoduše změnit jakoukoliv část návrhu. Zejména proto je OpenSCAD oblíbenou volbou pro návrh 3D objektů určených k 3D tisku.

Konstruktivní geometrie

Jednou z reprezentací 3D objektů v počítači je pomocí *konstruktivní geometrie těles*. V té je objekt popsán způsobem, který odráží postup, jaký by konstruktér použil při jeho návrhu. Z jednoduchých geometrických tvarů, kterým říkáme *geometrická primitiva*, je pomocí *množinových operací* a *prostorových transformací* vytvořen výsledný objekt.

Množinové operace jsou sjednocení, rozdíl a průnik. *Sjednocením* dvou množin získáme množinu, která obsahuje všechny prvky z obou množin. *Rozdílem* dvou množin rozumíme všechny prvky, které patří do první množiny, ale nepatří do druhé. *Průnik* dvou množin je množina společných prvků, tedy prvků, které patří do obou množin. Na obr. 1 jsou po řadě výsledky sjednocení, rozdílu a průniku krychle a koule.



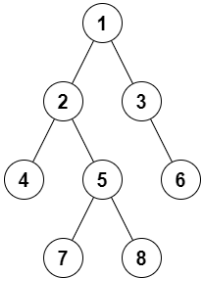
Obr. 1 Sjednocení, rozdíl a průnik krychle a koule

Příkladem prostorových transformací jsou posunutí v nějakém směru, otočení okolo některé z os v prostoru nebo změna velikosti. Velikost objektů lze změnit jejich prodloužením nebo zkrácením v jednom směru nebo jejich proporcionálním zvětšením či zmenšením.

[4] https://www.blender.org/

Metoda konstruktivní geometrie těles, anglicky constructive solid geometry, zkráceně CSG, je založena na reprezentaci tělesa stromovou strukturou tzv. *CSG stromem*, který uchovává dílčí kroky konstrukce.

Stromovou strukturu si lze zjednodušeně představit jako uspořádanou množinu prvků, kterým říkáme *uzly*. Mezi těmito uzly existuje hierarchický vztah potomek-rodič. Říkáme, že rodič je v hierarchii (uspořádání) nad potomkem (případně potomek je v hierarchii pod rodičem). Pokud uzel nemá žádného rodiče, nazývá se *kořenem stromu*. V každém stromu existuje právě jeden kořen. Prvky, které nemají žádného potomka, se nazývají *listy stromu*. Obr. 2 znázorňuje příklad stromu. Uzel označený číslem 1 je kořenem stromu, uzly 4, 6, 7 a 8 jsou listy stromu. Uzel 1 je rodičem uzlů 2 a 3. Uzel 2 má dva potomky a to uzly 4 a 5.



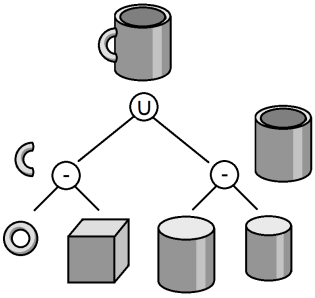
Obr. 2 Strom

Listy CSG stromu představují geometrická primitiva, ostatní uzly představují některou z operací nebo transformaci, která je aplikována na potomky uzlu. Těleso popsané CSG stromem je postupně konstruováno od listů směrem ke kořenu. Příklad CSG stromu je znázorněn na obr. 3. V tomto příkladu začínáme s torem,[5)] krychlí a dvěma válci, jedním větším a jedním menším. Od toru odečteme krychli (operace je označena -) a od většího válce odečteme menší. Výsledky těchto operací můžete vidět vedle uzlů. Tyto výsledné objekty sjednotíme (operace $\bigcup$). Výsledný objekt leží v kořenu stromu. Pro úplnost dodejme, že operace průniku, která se v našem příkladu nevyskytuje, se značí symbolem $\bigcap$.

Pro jednoduchost jsou ve stromu na obr. 3 vynechány prostorové transformace. Například jsme mohli začínat se dvěma stejnými válci a jeden z nich zmenšit. Dále v příkladu předpokládáme správné umístění toru

[5)] Torus je těleso, které vznikne otáčením kružnice kolem osy, která leží ve stejné rovině a nemá s ní společné body.

vzhledem ke krychli, kterou od něj odečítáme.



Obr. 3 CSG strom

OpenSCAD

OpenSCAD je nástroj pro tvorbu 3D modelů. Je zdarma dostupný na adrese `https://www.openscad.org` ve verzích pro operační systémy MS Windows, Linux a macOS. OpenSCAD slouží jako *překladač* (kompilátor), který čte textový popis tělesa a podle něj těleso vykresluje.

OpenSCAD je deklarativní programovací jazyk.[6)] Jeho dokumentaci nalezneme v [1]. OpenSCAD poskytuje dva základní způsoby tvorby 3D objektů. Jedním ze způsobů je již dříve zmíněná konstruktivní geometrie, druhým je extruze[7)] 2D tvarů do prostoru. 2D tvary opět můžeme popsat pomocí konstruktivní geometrie s tím, že v tomto případě jsou geometrickými primitivy základní geometrické tvary, jako například obdélníky, kružnice nebo mnohoúhelníky.

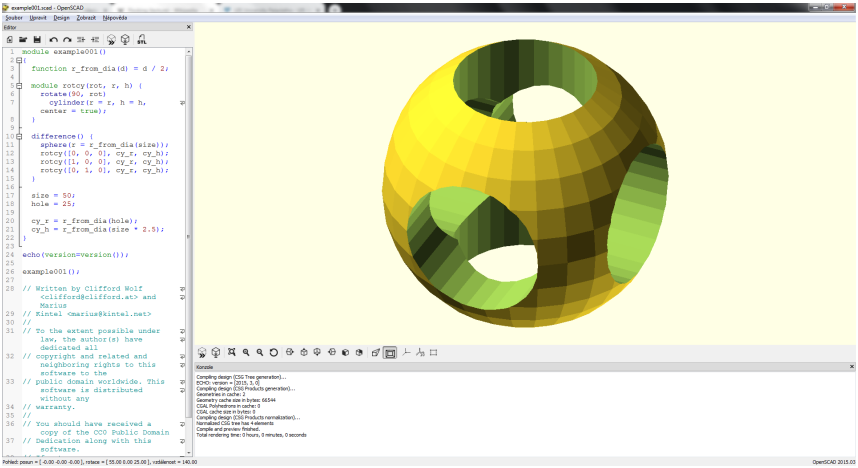
Prostředí OpenSCAD

Než začneme tvořit, seznámíme se s prostředím programu OpenSCAD, které je zobrazeno na obr. 4. V levé části okna se nachází *editor*, kde se zapisuje kód popisující těleso. Pravá část okna je rozdělena na dvě části. Horní část, představující virtuální trojrozměrný prostor, je *náhledová oblast* (viewing area), ve které se po přeložení kódu zobrazí modelované těleso. Body tohoto prostoru jsou zadávány v kartézské soustavě souřadnic. Kartézská soustava souřadnic je soustava souřadnic, u které jsou souřadné

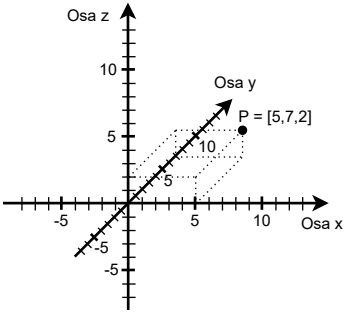
[6)] V jazyce popisujeme tvar, velikost a způsob kombinace jednotlivých částí, ne to, jak konkrétně toho dosáhneme.

[7)] Extruze je proces, při kterém se tvaruje materiál tím, že se vytlačuje skrz otvor, čímž vytváří objekty s konstantním průřezem.

osy vzájemně kolmé přímky, které se protínají v jednom bodě – počátku soustavy souřadnic. Na obr. 5 je znázorněn bod P v kartézské soustavě souřadnic. Tento bod má souřadnice $[5, 7, 2]$. Bod o souřadnicích $[0, 0, 0]$ nazýváme *počátek souřadnic*.



Obr. 4 Vývojové prostředí



Obr. 5 Kartézská soustava souřadnic

Pracovní plochu je možné za pomoci myši posouvat,[8] otáčet[9] a přibližovat/oddalovat ji.[10] V menu je možné si plochu přizpůsobit, například je

[8] Při stisku pravého tlačítka myši a jejím tažením.

[9] Při stisku levého tlačítka myši a jejím tažením.

[10] Pomocí kolečka myši.

možné si zapnout nebo vypnout zobrazení os a pravítka. Část pod náhledovou oblastí slouží pro výpis zpráv týkajících se překladu, chyb a varování, případně uživatelských výpisů.

Program

Program zapsaný v programovacím jazyce OpenSCAD představuje popis nějakého modelu. Základními stavebními kameny téměř každého programovacího jazyka jsou *hodnoty*, *výrazy* a *příkazy*. Hodnoty reprezentují informace (data), se kterými programy pracují. Například to mohou být čísla, textové řetězce a jiné. K jejich vytváření slouží výrazy. Příkazy dávají počítači instrukce, co má udělat, například vytvořit geometrické primitivum. Jednotlivé příkazy se v kódu oddělují středníkem (znakem `;`).

V první části si ukážeme pouze základní práci s jazykem OpenSCAD. Zejména se zaměříme na vytváření geometrických primitiv (`objekt`), aplikací geometrických transformací na objekty (`transformace`) a kombinování objektů do složitějších objektů pomocí množinových operací (`operace`). Jazyk OpenSCAD ovšem nabízí i složitější konstrukce běžné v jiných programovacích jazycích, jako jsou proměnné, konstrukce pro řízení toku dat (cykly a větvení), možnost tvorby vlastních funkcí, knihoven a jiné.

Pro vytváření objektů používáme příkazy, které se skládají ze jména (to většinou odpovídá názvu objektu, který chceme vytvořit), jenž je následované parametry určující vlastnosti vytvářeného objektu, například specifikujeme jeho velikost. Parametry píšeme do kulatých závorek přímo za název příkazu.

```
objekt(parametry);
```

Na vytvořené objekty je možné aplikovat geometrické transformace. U transformací je možné v kulatých závorkách specifikovat tuto transformaci (například o kolik se má objekt posunout, otočit či zvětšit nebo zmenšit). Na objekt je možné aplikovat více transformací, ty se píší za sebou a oddělují se mezerou.

```
transformace(parametry_transformace) objekt(parametry);
```

Objekty kombinujeme pomocí operací následovně:

```
operace(){
 objekt_1(parametry_1);
 ...
 objekt_n(parametry_n);
}
```

Za názvem operace jsou ve složených závorkách vypsány všechny objekty, na které se má operace aplikovat.

Výsledkem aplikace transformací a operací na objekty vznikají nové objekty, na které je možné aplikovat další transformace, případně je použít v dalších operacích. Konkrétní názvy objektů a operací spolu s parametry, které je možné zadat, si ukážeme v následujících kapitolách.

Mimo výše zmíněné prvky je možné do kódu psát i *komentáře*. Vše, co je na řádku za `//` nebo mezi `/*` a `*/`, je překladačem ignorováno.

Geometrická primitiva – objekty

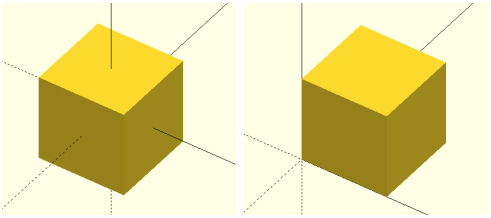
V této části si popíšeme základní geometrická primitiva, která používáme k vytváření modelů. Těmito primitivy budou krychle, koule, válec a mnohostěn.

Krychle

Krychli vytvoříme pomocí příkazu `cube()`. V kulatých závorkách lze specifikovat velikost krychle (parametr `size`) a také, kde se má vytvořit (parametr `center`). Parametry se oddělují čárkou a nezáleží na pořadí v jakém je uvádíme.

```
cube(size=1,center=false);
```

Parametr `center` může být nastaven na hodnotu `true` (pravda) nebo `false` (nepravda). Pokud je nastaven na `true`, pak je krychle vytvořena tak, že její střed leží v počátku souřadného systému. Pokud je nastavena na `false`, pak tam leží její vrchol. Na obr. 6 je vlevo výsledek krychle vytvořené s parametrem `center=true` a vpravo `center=false`.



Obr. 6 Krychle s parametrem `center=true` a krychle s parametrem `center=false`

Parametr `size` určuje velikost vytvářené krychle. Velikost může být zadána jedním číslem (v tom případě bude vytvořena krychle o zadané

délce stran), nebo trojicí čísel v hranatých závorkách `[x,y,z]`. Vytvořené těleso je pak kvádr s délkami stran x, y a z.

Ani jeden z parametrů není povinný a je možné jej vynechat. V takovém případě je použita *výchozí hodnota*. Výchozí hodnota parametru `size` je `1`, parametr `center` má výchozí hodnotu `false`. Následující příkazy vytvoří stejné krychle:

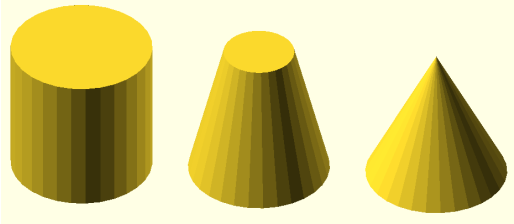
```
cube(size=[1,1,1],center=false);
cube(size=1,center=false);
cube(center=false,size=1);
cube(size=1);
cube();
```

Jak je vidět na předchozím příkladu, parametry můžeme zadávat v libovolném pořadí. Jsou-li hodnoty parametrů zadávány ve správném pořadí (jak určuje dokumentace), lze vynechat jejich název a zapsat jen hodnoty:

```
cube(1,false);
```

Válec

Válec se vytváří příkazem `cylinder()`. Při vytváření válce je třeba specifikovat jeho výšku (parametr `h`) a poloměr (parametr `r`) nebo průměr (parametr `d`)[11] podstavy. Místo parametru `r` je možné použít parametry `r1` a `r2`. Ty určují poloměr každé podstavy válce zvlášť.[12] Obdobně je možné nahradit `d` parametry `d1` a `d2`. Na obr. 7 můžeme vidět válec, komolý jehlan a jehlan vytvořené pomocí příkazu `cylinder()`. Jak vytvoříme pomocí příkazu `cylinder()` kužel? Nastavíme poloměr (průměr) jedné z podstav na 0.



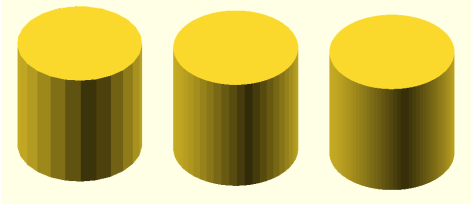
Obr. 7 Válec, komolý jehlan a jehlan

Pozici, kde se válec vytvoří, se stejně jako u krychle specifikuje parametrem `center`. Další parametry, kterými je možné ovlivnit vzhled válce,

[11] Pokud jsou předány oba parametry, pak je `d` ignorován.

[12] Je zřejmé, že se v tom případě nejedná o válec, ale o komolý kužel.

jsou parametry `$fa`, `$fs` a `$fn`. Všechny tyto parametry ovlivňují přesnost kruhové podstavy. Čím je jejich hodnota vyšší, tím je kruh přesnější.[13] Specifikuje se vždy jen jeden z nich.[14] Na obr. 8 můžeme vidět válce vytvořené s hodnotou parametru `$fn` nastavenou po řadě na 25, 50 a 100.



Obr. 8 Válec vytvořený s různými hodnotami parametru `$fn`

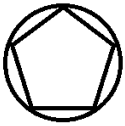
Pro vytvoření válce o výšce `1` a poloměru `1` použijeme příkaz:

```
cylinder(h=1,r=1,$fa=12);
```

Pokud zvolíme nízkou hodnotu parametru `$fn`, např.

```
cylinder(h=5,d=10,$fn=5);
```

vytvoří se pravidelný pětiboký hranol, který je vepsán do válce o průměru 10. Podstavu tohoto hranolu můžete vidět na obr. 9.



Obr. 9 Podstava pětibokého hranolu vepsaného do válce

Koule

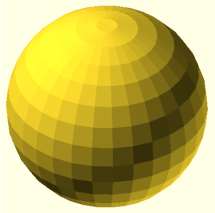
Kouli vytvoříme příkazem `sphere()`. Velikost koule je možné specifikovat poloměrem (parametr `r`) nebo zadáním velikosti jejího průměru (parametr `d`).[15] Koule o poloměru `1` se vytvoří příkazem

```
sphere(r=1);
```

[13] Parametr `$fa` představuje minimální úhel každého fragmentu, parametr `$fs` minimální délku oblouku a parametr `$fn` počet fragmentů v 360°.

[14] Pokud není zadán žádný z parametrů `$fa`, `$fs`, `$fn`, je použita výchozí hodnota parametru `$fa`, kterou je hodnota `12`, v důsledku toho výsledná podstava není kruhová.

[15] Oba parametry nelze použít zároveň. Pokud jsou nastaveny oba, je parametr `d` ignorován.



Obr. 10 Koule o poloměru 10

Pokud není zadán ani poloměr ani průměr, pak se vytvoří koule s poloměrem velikosti 1. Koule nemá parametr `center` a vždy se vytvoří se středem v počátku. Stejně jako u válce můžeme přesnost koule ovlivnit parametry `$fa`, `$fs` a `$fn`.

Mnohostěn

Nejobecnějším geometrickým primitivem v OpenSCAD je mnohostěn. Mnohostěn se vytváří příkazem `polygon()`. Je zadán množinou vrcholů (parametr `points`) a seznamem trojúhelníků (parametr `faces`),[16] které tvoří jeho povrch.

Každý vrchol je zadán trojicí `[x,y,z]` představující jeho souřadnice v trojrozměrném prostoru. Množina vrcholů je jejich výčet v hranatých závorkách.

```
points=[[x1,y1,y1],[x2,y2,z2],...,[xn,yn,zn]];
```

Každý trojúhelník je zadán trojicí vrcholů `[v1,v2,v3]`,[17] kde `v1`, `v2`, `v3` odkazují na vrcholy v množině `points`, kde jsou vrcholy očíslovány od `0` do `n-1` (`n` představuje počet vrcholů ve výčtu). Tomuto pořadí říkáme *index*. Trojice `[0,1,2]` představuje trojúhelník s vrcholy, jejichž souřadnice jsou uloženy ve výčtu vrcholů na indexech po řadě 0, 1 a 2, tedy první tři vrcholy. Mnohostěn se vytvoří příkazem

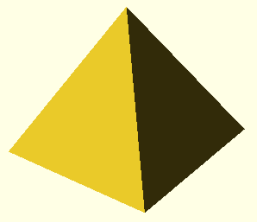
```
polyhedron(points=[...],faces=[...]);
```

Následující příkaz vytvoří čtyřboký jehlan, který je zobrazen na obr. 11.

```
polyhedron(points=[[5,5,0],[5,-5,0],[-5,-5,0],[-5,5,0],[0,0,10]], faces
    =[[0,1,4],[1,2,4],[2,3,4],[3,0,4],[1,0,3],[2,1,3]]);
```

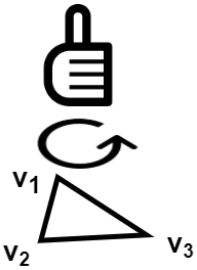
[16] Od verze 2014.03 to nemusí být jen trojúhelníky, ale obecné mnohoúhelníky. Je nutné, aby vždy všechny vrcholy mnohoúhelníku ležely v jedné rovině. Proto je lepší používat jen trojúhelníky, kde je tato vlastnost splněna vždy.

[17] V případě mnohoúhelníků se nejedná o trojice, ale obecně o n-tice.



Obr. 11 Jehlan vytvořený pomocí mnohostěnu

U objektů vytvářených pomocí `polyhedron()` je třeba si dávat pozor na pořadí vrcholů jednotlivých trojúhelníků (v parametru `faces`). Vrcholy trojúhelníku se při pohledu z venku na objekt zadávají proti směru hodinových ručiček. Můžeme také říct, že jsou uspořádány podle pravidla pravé ruky, které je znázorněno na obr. 12 a říká: pokud pokrčené prsty pravé ruky ukazují směr zadaných vrcholů, pak palec ukazuje směrem ven.



Obr. 12 Pravidlo pravé ruky

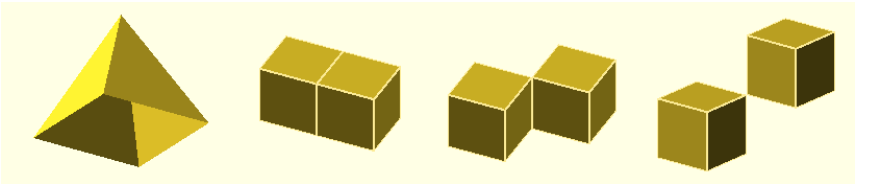
Další na co je třeba si dát pozor je, že vytvářený objekt musí být možné zkonstruovat. Konkrétně, modely musí být uzavřené, bez děr a otevřených vnitřků.[18] Modely nesmí obsahovat dotýkající se stěny, hrany a vrcholy.[19] Na obr. 13 jsou zobrazena tělesa, která nesplňují výše uvedené podmínky. První těleso obsahuje díru v podstavě,[20] další po řadě obsahují dotýkající se stěnu, hranu a vrchol.

[18] Této vlastnosti se říká, že jsou tělesa *watertight*.

[19] Tělesa neobsahující dotýkající se stěny, hrany a vrcholy se nazývají *manifoldy*.

[20] Tento výsledek obdržíme, pokud z předchozího příkladu odstraníme poslední dva trojúhelníky:

```
polyhedron(points=[[5,5,0],[5,-5,0],[-5,-5,0],[-5,5,0],[0,0,10]],
faces=[[0,1,4],[1,2,4],[2,3,4],[3,0,4]]);
```



Obr. 13 Chybně zadané mnohostěny

Transformace

Na vytvořené objekty je možné aplikovat prostorové transformace. Mezi základní transformace patří změna velikosti, otočení a posunutí.

Změna velikosti

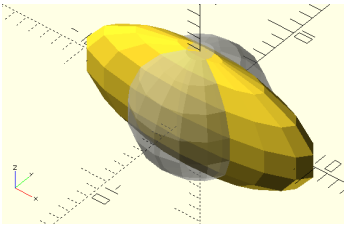
Velikost objektu lze v jazyce OpenSCAD změnit pomocí příkazů `scale()` a `resize()`. Příkaz `scale()` se používá následovně.

```
scale([x,y,z]) objekt();
```

`objekt()` v kódu představuje konkrétní objekt, například `cube()`. Předaný parametr `[x,y,z]` určuje, v jakém měřítku se změní velikost objektu ve směru os x, y a z. Konkrétně, aplikací

```
scale([2,0.5,1])
```

na nějaký objekt se tento objekt ve směru osy x dvojnásobně zvětší, v ose y se jeho velikost zmenší na polovinu a v ose z zůstane velikost nezměněna – viz obr. 14, kde byla operace aplikována na kouli o poloměru 5 zobrazené světle šedou barvou.



Obr. 14 Změna velikosti

Pokud chceme objektu změnit velikost na konkrétní hodnotu, použijeme příkaz `resize()` následovně:

```
resize([x,y,z],auto=true) objekt();
```

Trojice `[x,y,z]` v tomto případě označuje konkrétní velikost objektu ve směru jednotlivých os. Pokud je nějaká z těchto hodnot rovna nule a je nastaven parametr `auto` na `true`, bude objektu v tomto směru nastavena hodnota tak, aby byl zachován poměr zvětšení nebo zmenšení.[21)] Pokud není `auto=true`, pak velikost tělesa ve směrech s nulovou hodnotou zůstává stejná.

Otočení

Objekt je možné otočit pomocí příkazu `rotate()`. Otáčet je možné kolem všech os o stejný úhel, nebo okolo každé jinak. Použití je následující.

```
rotate(a,[x,y,z]) objekt();
rotate([x,y,z]) objekt();
```

V prvním případě parametr `a` představuje úhel ve stupních a druhý parametr `[x,y,z]` je trojice jedniček a nul, které představují okolo kterých os se má objekt otočit o úhel `a` (otáčí se okolo těch, které jsou rovny 1).

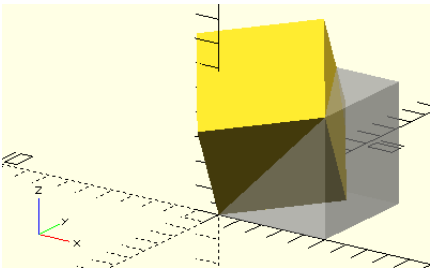
Druhé možné použití je takové, že se příkazu `rotate()` předává trojice úhlů, o kolik se má objekt otočit v každé z os. Příkaz

```
rotate(45,[1,0,1]) cube(size=1);
```

otočí krychli o 45° okolo os x a z (viz obr. 15). Příkaz

```
rotate([90,0,45]) cube(size=1);
```

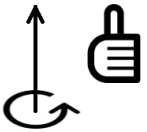
otočí krychli kolem osy x o 90° a okolo osy z o 45°.



Obr. 15 Otočení

[21)] Pokud jsou zadaná dvě různá zvětšení, třetí se upraví podle většího z nich.

Otáčení okolo osy probíhá podle pravidla pravé ruky. Pokud palec ukazuje v kladném směru osy kolem které otáčíme, pak pokrčené prsty ukazují směr rotace, jak ukazuje obr. 16.



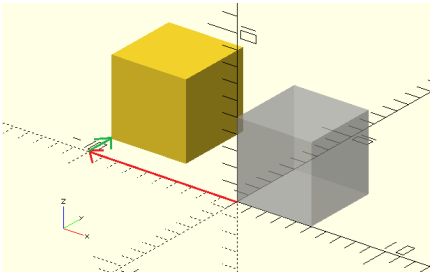
Obr. 16 Pravidlo pravé ruky

Posunutí

Pro posun objektu použijeme příkaz `translate()`, jehož použití je následující:

```
translate([x,y,z]) objekt();
```

Čísla `x`, `y` a `z` vyjadřují o kolik se má posunout objekt ve směru os x, y a z. Na obr. 17 můžeme vidět krychli posunutou o 10 ve směru osy x, o 2 ve směru osy y a o 0 ve směru osy z.



Obr. 17 Posunutí

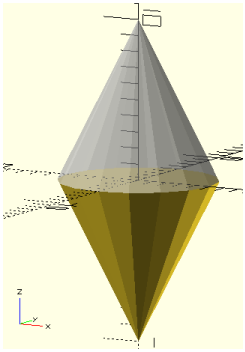
Zrcadlení

Kromě výše zmíněných transformací, můžeme objekt zrcadlit dle nějaké roviny pomocí transformace `mirror()`, jehož použití je následující:

```
mirror([x,y,z]) objekt();
```

Čísla `x`, `y` a `z` určují bod v prostoru, ze kterého je vedena přímka k počátku souřadného systému. Objekt je pak zrcadlen dle roviny kolmé k této přímce v počátku. Na obr. 18 můžeme vidět jehlan zrcadlený dle XY roviny pomocí příkazu

```
mirror([0,0,1]) cylinder(r1=5, r2=0, h=10);
```



Obr. 18 Zrcadlení

Skládání transformací

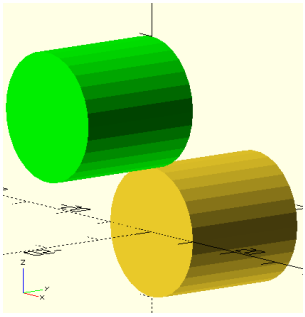
Na objekt je možné aplikovat více transformací. Jednotlivé transformace se oddělují mezerou. V zápisu záleží na pořadí, ve kterém jsou operace zapsány. Operace se vykonávají v opačném pořadí, než jsou uvedeny, tedy zprava doleva. Např.

```
translate([0,20,0]) rotate([90,0,0]) cylinder(h=20,r=10);
```

nejprve otočí válec o 90° kolem osy x a pak jej posune ve směru osy y o 20 jednotek (na obr. 19 žlutý válec).

```
rotate([90,0,0]) translate([0,20,0]) cylinder(h=20,r=10);
```

V tomto příkazu je nejprve válec posunut ve směru osy y a pak teprve otočen (na obr. 19 zelený válec).



Obr. 19 Skládání transformací

Operace

V předchozích částech jsme se věnovali příkazům vytvářejícím jednoduchá geometrická primitiva a možnostem, jak změnit jejich polohu a velikost pomocí geometrických transformací. Nyní si ukážeme, jakým způsobem kombinovat více těles dohromady a vytvářet tak složitější tvary pomocí množinových operací sjednocení, průnik a rozdíl.

Pro sjednocení v OpenSCADu používáme příkaz `union()`, který se aplikuje na objekty, jež jsou uvedeny ve složených závorkách za tímto příkazem. Pro sjednocení koule a krychle použijeme následující konstrukci.

```
union(){
 cube(size=10,center=true)
 sphere(d=13);
}
```

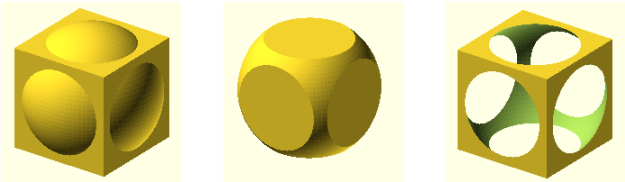
Pro průnik objektů se používá příkaz `intersection()`, obdobně, jako se používá `union()`.

```
intersection(){
 cube(size=10,center=true)
 sphere(d=13);
}
```

Pro rozdíl se používá příkaz `difference()`. Ten se aplikuje tak, že od prvního objektu, který je uveden ve výčtu ve složených závorkách, se odečítají všechny další objekty. Konkrétní příklad vypadá následovně.

```
difference(){
 cube(size=10,center=true)
 sphere(d=13);
}
```

Výsledky sjednocení, průniku a rozdílu krychle a koule z předchozích příkladů můžete vidět na obr. 20.



Obr. 20 Sjednocení, průnik a rozdíl krychle a koule

Dvourozměrné objekty

V této části se budeme věnovat dvourozměrným objektům a ukážeme si, jak z nich vytvořit objekty trojrozměrné. Dvourozměrné objekty jsou objekty, které jsou nekonečně tenké, vytváří se v rovině os x a y (XY *rovina*) a v náhledu (po stisku F5) se zobrazují s tloušťkou 1. Tyto objekty se vytváří pomocí klíčového slova, které označuje typ objektu, a v kulatých závorkách se specifikují vlastnosti těchto objektů stejně, jak tomu bylo v případě trojrozměrných objektů.

Mezi základní dvourozměrné objekty patří čtverec, který se vytváří příkazem `square()` (viz obr 21a). Parametry tohoto příkazu jsou stejné jako u krychle, tedy `size` (velikost), která může být buď jedno číslo, nebo dvojice v hranatých závorkách,[22)] a `center`, ovlivňující umístění čtverce.

```
square(size=1,center=false);
```

Dalším základním objektem je kruh, který vytvoříme příkazem `circle()` (viz obr 21b). Parametry, které mu můžeme zadat, jsou obdobné jako u koule, tedy můžeme specifikovat jeho velikost parametry `r` (poloměr) nebo `d` (průměr) a přesnost ovlivnit parametry `$fa`, `$fs` a `$fn`.

```
circle(r=1);
```

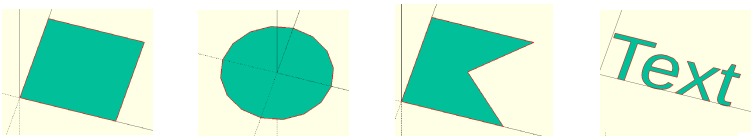
Nejobecnějším dvourozměrným objektem je mnohoúhelník. Příkaz vykreslující mnohoúhelník je `polygon()` (viz obr 21c). Jako parametry se mu předávají vrcholy (`points`), které jsou zadávány dvojicí `[x,y]` představující x-ovou a y-ovou souřadnici bodu, a cesty (`paths`), což jsou sekvence indexů (indexy v hranatých závorkách) odkazujících do množiny bodů představující hranice mnohoúhelníku. Pokud není cesta zadána, pak je hranice dána pořadím vrcholů v parametru `points`. Cest obecně může být více (v parametru `path` je jejich výčet). První cesta představuje vnější hranici mnohoúhelníku, všechny další jsou chápány jako díry v něm. Mnohoúhelník musí být vždy uzavřený, což znamená, že poslední vrchol je roven prvnímu a je doplněn automaticky. Níže uvedený příkaz vykreslí čtverec se stranou délky 10:

```
polygon(points=[[0,0],[0,10],[10,0],[10,10]], paths=[[0,1,3,2]]);
```

Dalším dvourozměrným objektem je objekt text, který vytvoříme příkazem `text()` (viz obr 21d). Parametr `text` udává text, který chceme zobrazit. Vzhled textu je možné měnit dalšími parametry.

[22)] V takovém případě se nejedná o čtverec, ale o obdélník.

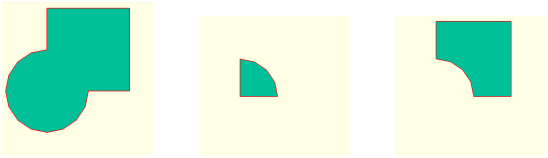
```
import(file="smiley.dxf");
```



(a) Čtverec (b) Kruh (c) Mnohoúhelník (d) Text

Obr. 21 2D objekty

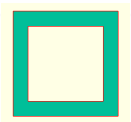
Na dvourozměrné objekty můžeme aplikovat transformace a množinové operace obdobně jako na trojrozměrné objekty.[23)] Na obr. 22 vidíme po řadě výsledky sjednocení, průniku a rozdílu čtverce a kruhu.



Obr. 22 Sjednocení, průnik a rozdíl čtverce a kruhu.

Navíc můžeme objekty zmenšovat a zvětšovat pomocí příkazu `offset()`, kterému jako parametr zadáváme `r` nebo `delta`, které určují o kolik se má objekt zmenšit (záporné číslo) nebo zvětšit (kladné číslo). Rozdíl mezi těmito parametry je v tom, že při použití parametru `r` budou rohy výsledného objektu zakulacené, v případě parametru `delta` budou ostré. Na obr. 23 vidíme výsledek následujícího kódu.

```
difference(){
    offset(delta=2) square(10);
    square(10);
}
```



Obr. 23 Offset

Vytvoření 3D objektu z 2D

Dvourozměrné objekty samy o sobě nemůžeme využít při konstrukci trojrozměrných modelů. Musíme z nich nejprve vytvořit trojrozměrné objekty. K tomu můžeme využít tzv. *extruzi*. Tento pojem by se dal přeložit

[23)] Transformace jako jsou změna velikosti a posun je možné provádět jen ve směrech os x a y a rotace má smysl jen kolem osy z.

do českého jazyka jako vytlačení. V OpenSCADu existují dva typy extruze a to lineární a rotační.

Lineární extruze

Lineární extruze se provádí příkazem `linear_extrude()`, který vyžaduje parametr `height` a aplikuje se na dvourozměrný objekt následovně:

```
linear_extrude(height=1) objekt();
```

Příkaz vytvoří trojrozměrný objekt, který bude mít podstavy rovny objektu `object()` a jeho výška bude dána předaným parametrem `height`.

Následující příkaz vytvoří ze čtverce o straně 12 jednotek kvádr o výšce 10 jednotek. Výsledek vidíme na obr. 24a.

```
linear_extrude(height=10) square(size=12,center=true);
```

(a) Lineární extruze (b) Parametr `scale` (c) Parametr `twist`

Obr. 24 Lineární extruze

Extruze nemusí být pravidelná. Pokud použijeme v `linear_extrude()` parametr `scale`,[24] pak se změní měřítko horní podstavy v poměru zadaném tímto parametrem.

```
linear_extrude(height=1,scale=1) objekt();
```

Dolní podstava trojrozměrného objektu bude odpovídat objektu `object()`, horní podstava bude zvětšena `scale` krát.[25] Extruze se sbíhá směrem k ose z (rozbíhá od osy z). V následujícím příkladu vidíme výsledek při použití parametru `scale`, který nastavíme na jednu polovinu. Výsledek následujícího kódu je zobrazen na obr. 24b.

```
linear_extrude(height=10,scale=0.5) square(size=12,center=true);
```

Dalším parametrem ovlivňujícím extruzi je parametr `twist`. Ten představuje počet stupňů, o kolik se objekt během extruze otočí (otáčí se po

[24] Výchozí hodnota tohoto parametru je rovna 1. Proto pokud tento parametr nenastavíme, pak se velikost objektu během extruze nezmění.

[25] Během extruze se velikost objektu postupně mění.

směru hodinových ručiček, pokud chceme otáčet proti směru, použijeme zápornou hodnotu).[26)]

```
linear_extrude(height=1,twist=0)
```

I v tomto případě probíhá otáčení kolem osy z. Následující kód vytvoří objekt, který je zobrazen na obr. 24c.

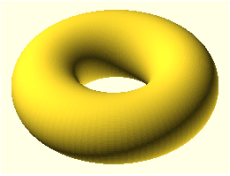
```
linear_extrude(height=10,twist=90) square(size=12,center=true);
```

Rotační extruze

Příkaz `rotate_extrude()` vytvoří trojrozměrný objekt tak, že otočí dvourozměrný objekt kolem osy z.[27)]

```
rotate_extrude(angle=360,$fa=12) objekt();
```

Parametr `angle` určuje o kolik stupňů se má objekt otočit po směru hodinových ručiček. Pokud není zadán úhel, otočí se o 360°. Parametr `$fn` ovlivňuje přesnost vykreslení objektu. Například torus (viz obr. 25) získáme rotační extruzí kruhu následovně.



Obr. 25 Rotační extruze

```
rotate_extrude(angle=360,$fn=100) translate([2,0,0]) circle(r=1);
```

Závěr

V tomto článku jsme ukázali základy OpenSCAD, od tvorby jednoduchých geometrických tvarů po jejich transformace. Řekli jsme, jak pomocí množinových operací tyto geometrické tvary skládat. V dalším díle se zaměříme na pokročilejší techniky, jako je využívání proměnných, cyklů a větvení, abychom mohli tvořit ještě složitější 3D modely. Také popíšeme, jak vytvářet znovupoužitelné části kódu, což nám usnadní práci a urychlí proces tvorby modelu.

Literatura

[1] https://openscad.org/documentation.html

[2] Žára, J., Beneš, B., Sochor, J., Felkel, P.: Moderní počítačová grafika. Computer Press, 2004.

[26)] Výchozí hodnota tohoto parametru je rovna 0.

[27)] Objekty leží v rovině XY. Těleso vzniklo otočením objektu kolem osy y a pak bylo otočeno tak, jako by vzniklo otočením kolem osy z.