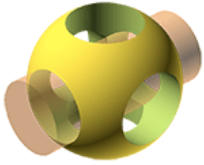
KATEDRA
INFORMATIKY
UNIVERZITA PALACKÉHO V OLOMOUCI

Parametrické modelování

KMI/3DT 3D tisk

Mgr. Markéta Trnečková, Ph.D.
www.marketa-trneckova.cz

OpenSCAD



OpenSCAD

https://www.openscad.org

3D objekty

- `cube(size,center)`
- `sphere(r|d,$fa,$fs, $fn)`
- `cylinder(h,r|d,center)`
- `cylinder(h,r1|d1,r2|d2, center)`
- `polyhedron(points,faces,convexity)`

2D objekty

- `square(size,center)`
- `square([width,height],center)`
- `circle(r|d,$fa,$fs, $fn)`
- `polygon([points])`
- `polygon([points],[paths])`
- `text(t, size, font, halign, valign, spacing, direction, language, script)`
- `import("")`

Transformace

- `scale([x, y, z])`
- `resize([x, y, z], auto)`
- `rotate(a, [x, y, z])`, `rotate([x, y, z])`
- `translate([x, y, z])`
- `mirror([x, y, z])`
- `multmatrix(m = [ ... ])`
- `offset(r|delta,chamfer)`
- `hull()`
- `minkowski()`
- `linear_extrude(height,center,twist,slices)`
- `rotate_extrude(angle)`
- `surface(file)`

Operace

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

Vykreslení

- F5 - náhled
- F6 - vyrenderování
- Export jako STL

Ladění

- `color("red")`, `color([r,g,b])`, `color([r,g,b,a])`
- `#`
- `%`
- `!`
- `*`

Hodnoty

- Dynamicky typovaný
- Pevná množina datových typů, které nejsou pojmenované:
  - čísla – `42`, `0.25`
  - logické hodnoty – `true`/`false`
  - řetězce – `"text"`
  - rozsah (range) – `[0 : 1 : 10]`
  - nedefinovaná hodnota – `undef`

Hodnoty
Čísla

- 64 bitové číslo s plovoucí desetinnou čárkou (některá čísla nejde vyjádřit přesně)
- Decimální notace

Příklad

```
42
-1
0.5
1.234e+5
```

- Pojmenovaná čísla – `PI`
- Při operacích je možné získat `nan` (toto ale není konstanta)

Hodnoty
Čísla

- aritmetické operace: + (sčítání), – (odčítání), * (násobení) a / (dělení)
- goniometrické operace: `cos()`, `sin()`, `tan()`, `acos()`, `asin()`, `atan()`
- další: `abs()` (absolutní hodnota), `ceil()` (zaokrouhlení nahoru), `round()` (zaokrouhlení k nejbližší celé hodnotě), `floor()` (zaokrouhlení dolů), `sqrt()` (odmocnina)

Hodnoty
Logické hodnoty

- `true` a `false`
- logické operace: `&&` (and), `||` (or) a `!` (not)
- nepravda: prázdný řetězec (`""`), prázdný vektor (`[]`) a kladná i záporná nula (`0` a `-0`)
- pravda: ostatní

Hodnoty
Řetězce

- řetězec = sekvence (unicode) znaků v uvozovkách
- zejména se využívají k výpisům – funkce `echo()`
- speciální znaky: `\n`, `\t`
- `\uxxxx` nebo `\Uxxxxxx`, kde `x` reprezentují čtyř nebo šestimístný unicode kód

Příklad

```
echo("\U01f436 jí \U01f9b4.");
```

ECHO: "🐶 jí 🦴."

Hodnoty
Řetězce

- Funkce pro práci s řetězci:
  - `str()` – převede argumenty na řetězec a ty spojí do jednoho

Příklad

Vyzkoušejte následující příklad:

```
cislo = 4;

echo("Toto je", cislo, 2);
echo(str("Toto je", cislo, 2));
```

  - `len()` – délka řetězce

Hodnoty
Vektory

- vektor = kolekce hodnot libovolných typů, výrazů (které vyhodnotí jako hodnota)
- v hranatých závorkách (`[` a `]`) oddělené čárkami

Příklad

```
v = [3, 5, [6,7], [[8,9],[10,[11,12],13]], "string"];
```

- délka vektoru: `len()`
- spojení vektorů: `concat()`
- přístup k prvkům – přes indexy v hranatých závorkách
- indexujeme od `0`
- překročení mezí vektoru – `undef`

Hodnoty

Vektory

Příklad

Pro následující vektor

```
v = [ [1], [], [2,3,4], "text", "x", [[5,6],[7,8,9],[[10,11],[12]]] ];
```

určete následující prvky a jejich délku.

```
v[0]
v[1]
v[5]
v[5][1]
v[5][2]
v[5][2][0]
v[5][2][0][1]
```

Hodnoty
Vektory

- vektory zejména používáme pro specifikaci v jednotlivých osách
- `v.x` = `v[0]`
- `v.y` = `v[1]`
- `v.z` = `v[2]`

Hodnoty
Rozsah

- využití zejména v cyklech
- `[start:konec]`
- `[start:krok:konec]`
- není to vektor!

Příklad

Podívejte se jak se pomocí `echo()` vypíše rozsah.

Proměnné

- identifikátor – malé a velké znaky anglické abecedy, čísla a podtržítko
- speciální proměnné – identifikátory začínají znakem `$`
- přiřazení hodnoty: `promenna = hodnota;`
- OpenSCAD je funkcionální – proměnná má v rámci své platnosti jednu hodnotu
- proměnné je hodnota nastavena při překladu, ne za běhu programu

Příklad

Zkuste následující kód přeložit v OpenSCAD. Než to ale uděláte, zamyslete se nad tím, jaký bude výsledek.

```
a = 3;
echo(a);
a = 4;
echo(a);
```

Proměnné

- Je možné přiřazovat:
  - hodnoty
  - hodnoty, které jsou výsledkem nějaké operace
  - hodnoty jiných proměnných

Příklad

Vysvětlete proč následující konstrukce není v OpenSCAD v pořádku: `x = x + 1;`

Proměnné

- Je možné přiřazovat:
  - hodnoty
  - hodnoty, které jsou výsledkem nějaké operace
  - hodnoty jiných proměnných
- Proměnná, které nebyla přiřazena hodnota, má hodnotu `undef`

```
echo("Promenna a ma hodnotu ", a);
```

Příklad

Vysvětlete proč následující konstrukce není v OpenSCAD v pořádku: `x = x + 1;`

Proměnné

Platnost proměnných

- lokální rozsah (scope)
- složené závorky vytváří nový scope uvnitř současného

Příklad

```
a = 3;
echo(a);
translate([0, 0, 0]){
    a = 4;
    echo(a);
}
echo(a);
```

Proměnné
Platnost proměnných

- samotné závorky nový scope netvoří!

Příklad

```
a = 3;
echo(a);
{
    a = 4;
    echo(a);
}
```

Cykly
for

- `for(i=[0:n]) { ... }`
- cyklus – přes hodnoty rozsahu, nebo vektoru
- pro každou hodnotu se vytvoří nový scope

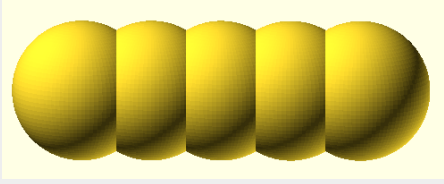
Příklad

Kolikrát proběhne cyklus, pokud bude `i = [1,n]` ?

Cykly
for

Příklad

Vytvořte sérii n koulí:

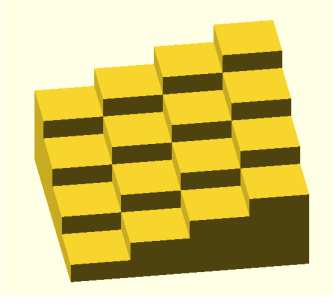


Cykly

Vnořené cykly

Příklad

```
for (i=[0:3]){
    for (j=[0 : 3]){
        translate([3*i,3*j,0]) cube([3,3,i+j+1]);
    }
}
```



Cykly
Vnořené cykly

Ekvivalentní zápis:

Příklad

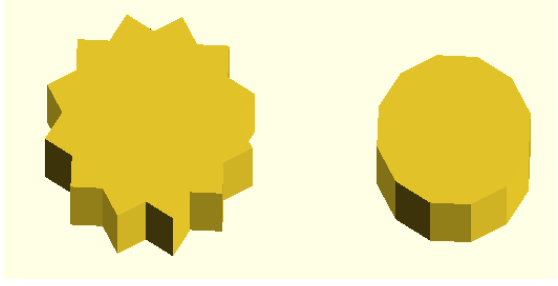
```
for (i=[0:3], j=[0 : 3]){
    translate([3*i,3*j,0]) cube([3,3,i+j+1]);
}
```

Příklad

Porovnejte s chováním jazyka C.

Cykly
intersection_for

- `intersection_for(i=[0:n]) { ... }`



Větvení

- `if(podminka) { ... }`
- `else { ... }`
- `<, <=, >, >=, ==, !=`
- `!, &&, ||`

Příklad

Co vypíše následující kód?

```
b = -5;
echo(b);
if (b < 0){
    b = 5;
    echo(b);
}
echo(b);
```

Větvení

- `test ? pravda : nepravda`

Příklad

```
abs_a = a < 0 ? -a : a;
```

Funkce

- `function jmeno(parametry) = hodnota;`
- parametrům můžeme přiřadit defaultní hodnoty

Příklad

```
function f(x = 3) = x + 1;
echo(f());
echo(f(6));
```

Funkce

Příklad

```
function funkce(x, y = 5) = x - y;

echo(funkce());  // chyba, protoze x je undef
echo(funkce(2));  // -3 x = 2, y = 5
echo(funkce(2, 2));  // 0
```

V následujícím kódu není možné volat funkci jen s jedním argumentem. **Proč?**

Příklad

```
function funkce(x = 5, y) = x - y;

echo(funkce());  // chyba, protoze y je undef
echo(funkce(y = 2));  // 3 x = 5, y = 2
echo(funkce(y = 2, x = 3));  // 1 x = 3, y = 2
```

Funkce

Příklad

Vytvořte funkci, která ze zadaných vstupních parametrů - start, počet prvků a konec vytvoří vektor začínající startem, končící koncem a mající zadaný počet prvků.

Příklad

Vytvořte funkci, která pro zadané číslo vrátí -1 pokud je číslo záporné, jinak vrátí 1.

Funkce

Funkce může používat i jiné proměnné, než které jí jsou předány

Příklad

```
function funkce1(x) = x - y;
function funkce2(x) = x - z;

z = 3;
echo(funkce1(5));  // chyba, protoze y je undef
echo(funkce2(5));  // 2 x = 5, z = 3
echo(funkce2(5, z = 5));  // 0 x = 5, z = 5
```

Funkce
Rekurzivní funkce

Příklad

Jak udělat koncovou podmínku?

Funkce

Rekurzivní funkce

Příklad

```
function faktorial(x) = x==1 ? 1 : x*faktorial(x-1);
```

Moduly

```
module nazev (parametry){
    // telo modulu
}
```

- objektové moduly
- Moduly představující operace

Moduly

Objektové moduly

Příklad

```
module box (a=5){
    cube(size = a, center=true);
}


box();
box(a = 10);
translate([1,1,1]) box(1);
```

Moduly

Moduly představující operace

- `children()`

Příklad

```
module posun_z(z = 0){
    translate([0, 0, z]) children();
}

posun_z(10) cube(1);


posun_z(10){
    cube(10);
    sphere(5);
}
```

Moduly

Moduly představující operace

- počet potomků – `$children`
- přístup k jednotlivým potomkům – pomocí indexů
- indexy od 0 do `$children` -1

Příklad

```
children();          // vsichni potomci
children(index);     // jeden konretni potomek
children(rozsah);    // vsichni potomci v rozsahu indexu
children(vektor);    // potomci s indexy ve vektoru
```

Moduly

Příklad

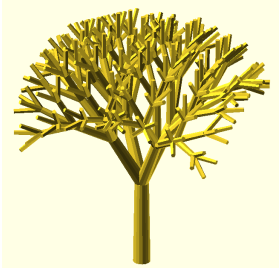
Vytvořte modul, který po aplikaci na objekty tyto objekty postupně posune vždy o 10 ve směru osy x.
První posune o 10, druhý o 20 a tak dál.

Moduly
Rekurzivní moduly

Příklad

```
module strom(velikost, n){
    if (n > 0){
        cylinder(r1=velikost/10, r2=velikost/12, h=velikost);
        translate([0,0,velikost])
            for(iter = [0 : 4]){
                rotate([30,0,iter*90])
                    strom(0.75*velikost, n-1);
            }
    }
}

strom(50,  5);
```



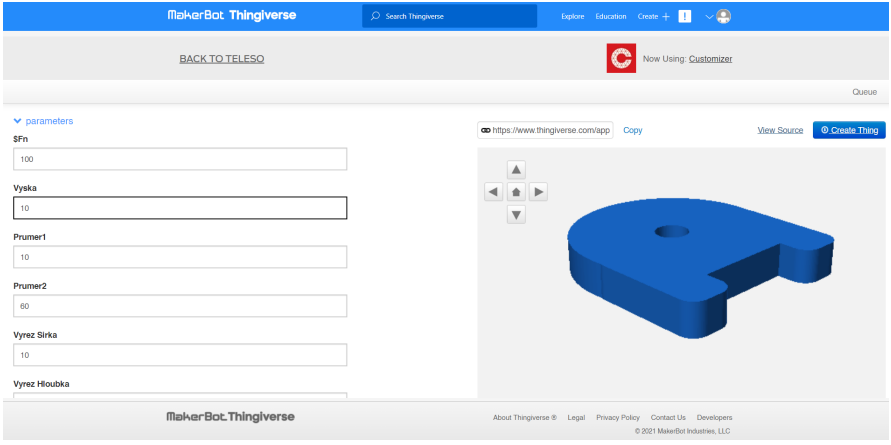
Knihovny

- `include <...>;`
- `use <...>;`

Příklad

```
use <MCAD/lego_compatibility.scad>;
block(1,2,1,reinforcement=true);
```

Thingiverse Customizer



Příklad

Vytvořte modul `teleso`, který vygeneruje těleso z dřívějších hodin (s tím, že je možné měnit velikosti).