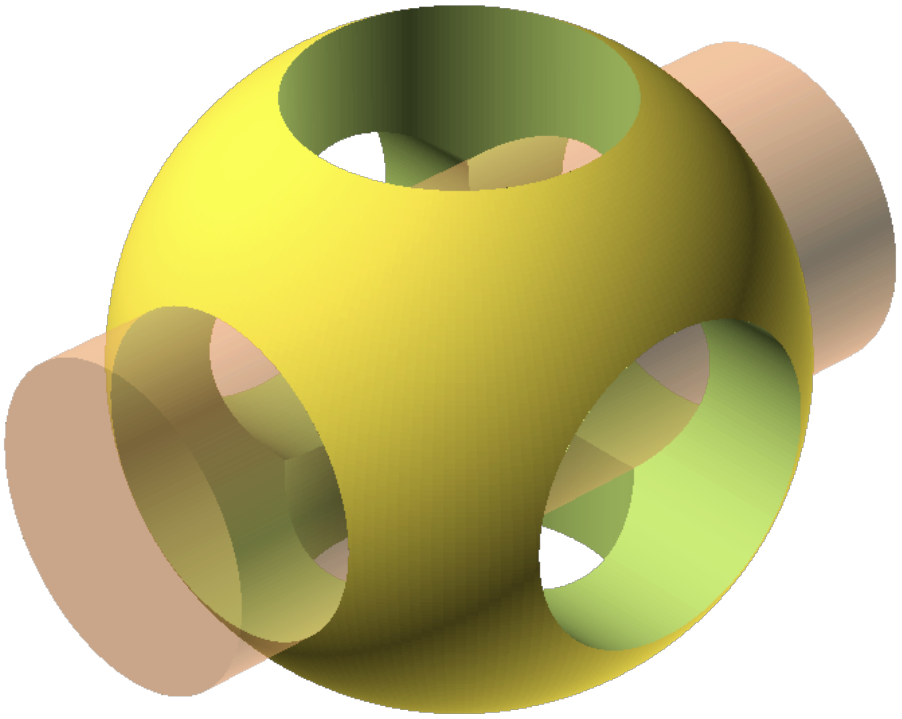
Premiers pas sur OpenSCAD

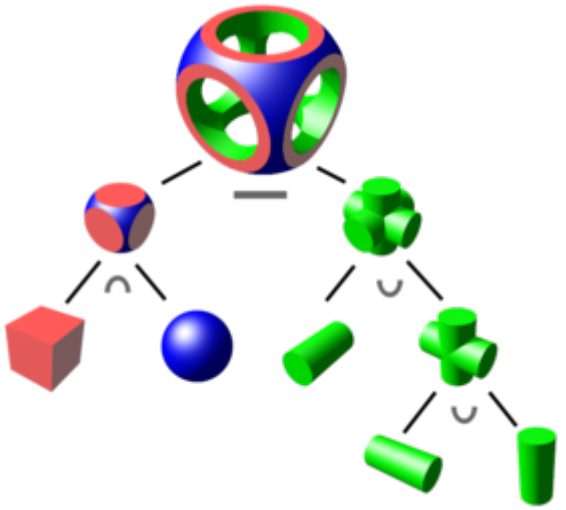


Auteur Marc Forner pour SoFAB

Introduction

késako

- OpenSCAD est un logiciel de modélisation 3D, orienté CAO
- La version nightly est une béta qui offrent des fonctionnalités intéressantes.
- Utilise la **géométrie de construction de solide** *(Constructive Solid Geometry ou CSG en anglais)*



Introduction

géométrie de construction de solides

« Cette technique de modélisation géométrique concerne la représentation d'un objet solide comme combinaison d'objets solides simples (exemple : cylindre, sphère, cône, tore, etc.) à l'aide d'opérateurs géométriques booléens (exemple : union, intersection, soustraction). » wikipédia

On peux résumer la modélisation sous OpenSCAD par 3 étapes :

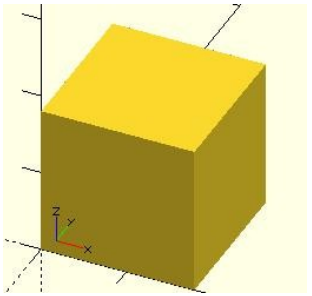
Primitives → Transformations → Opérations booléennes

Les primitives 3D

- Cuboïdes
- Spheres
- Cylindres
- Polyhedres

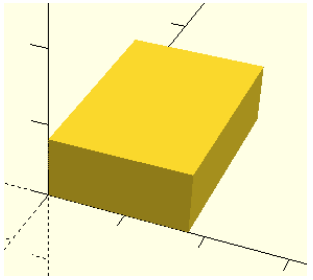
Les primitives 3D

cuboïdes



```
cube(18) ;
```

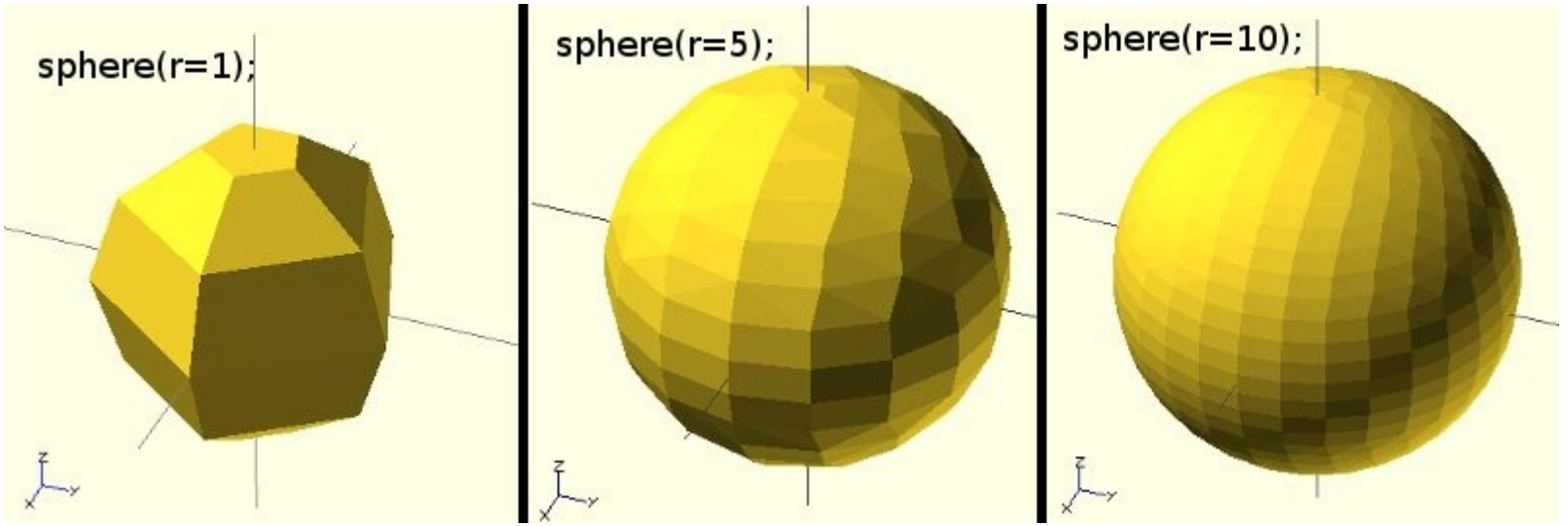
Tout ce qui est cartésien doit être mis sous crochets



```
cube([18,28,8]);
```

Les primitives 3D

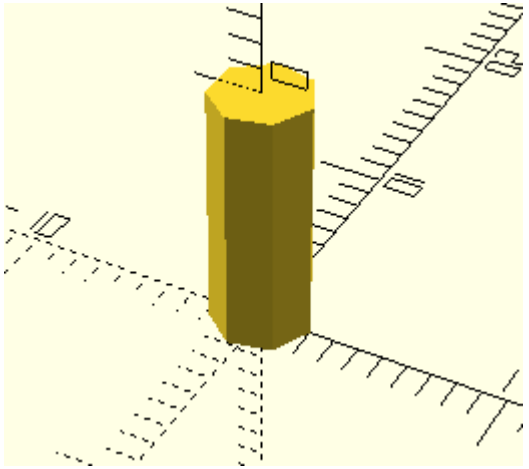
sphères

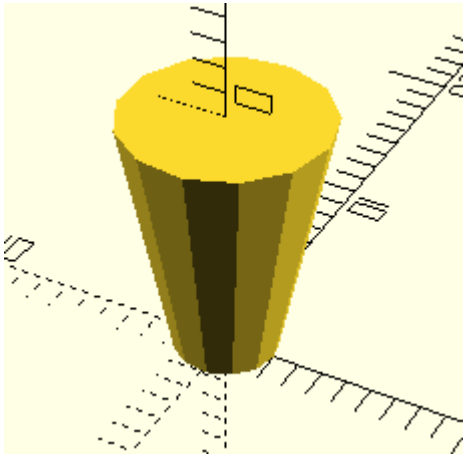


sphere(1) ; sphere(5) ; sphere(10) ;

Les primitives 3D

les cylindres

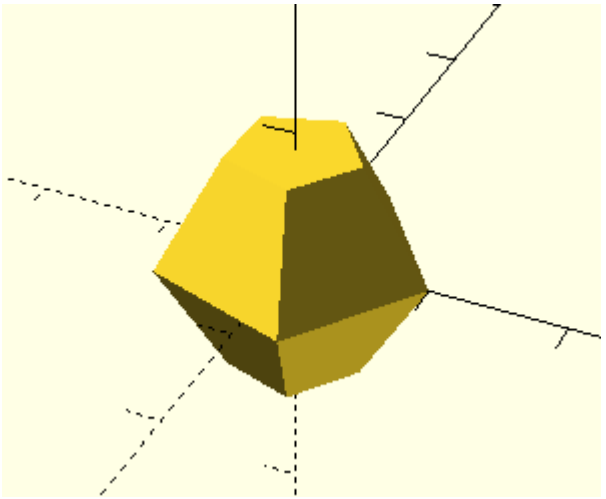


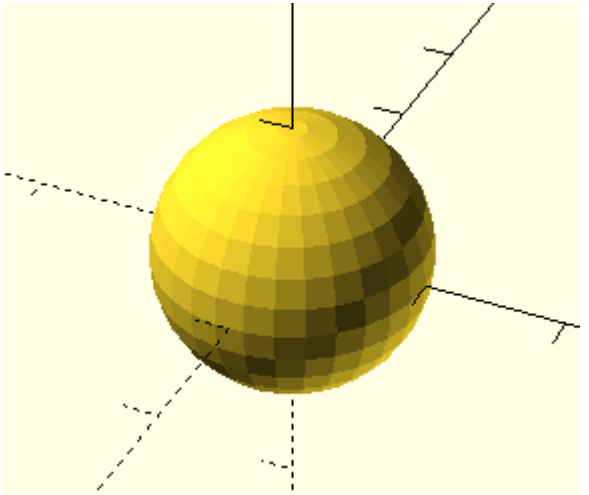


cylinder(r=2,h=10); cylinder (r1=2,r2=4,h=10);

Les primitives 3D

la fonction $fn





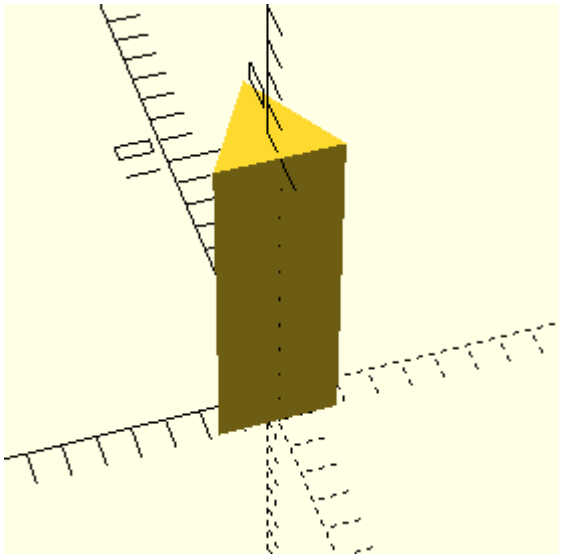
sphere(1);

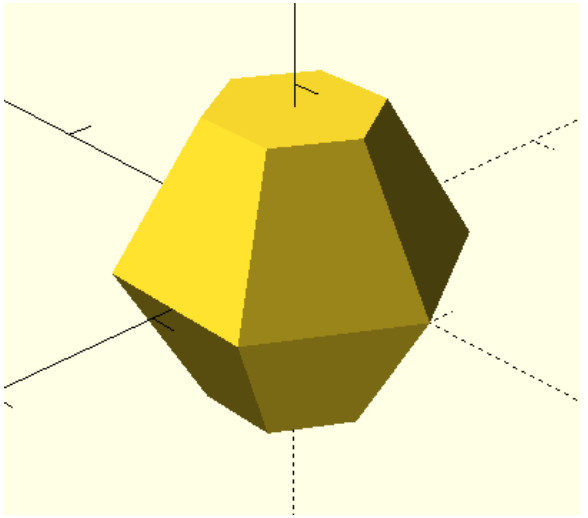
sphere(1,$fn=30);

Les primitives 3D

usage des primitives et $fn

$fn sert à lisser mais aussi a obtenir d'autres formes



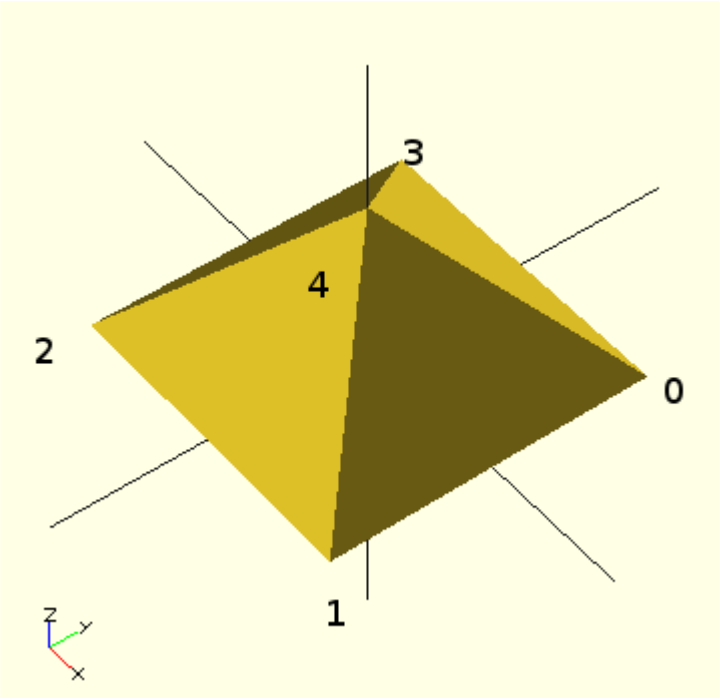


cylinder(r=2,h=10,$fn=3);

sphere(10,$fn=6);

Les primitives 3D

le polyhèdre



Les commentaires de lignes commencent par //

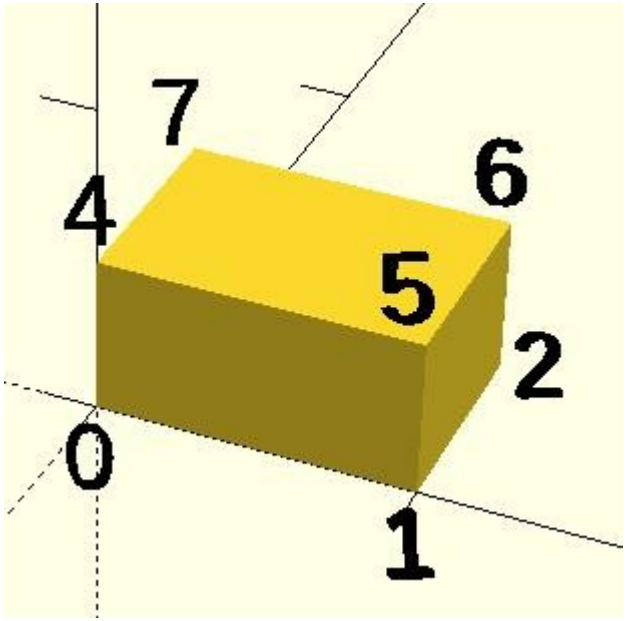
```
polyhedron(
  points=[ [10,10,0],[10,-10,0],[-10,-10,0],[-10,10,0],
           [0,0,10]  ],
  faces=[ [0,1,4],[1,2,4],[2,3,4],[3,0,4],
            [0,1,2,3] ]
 );
```

// Les 4 points de la base
// l'apex
// les 4 cotés triangulaires
// la base

Les primitives 3D

le polyhèdre

```
CubePoints = [
  [  0,  0,  0 ],  //0
  [ 10,  0,  0 ],  //1
  [ 10,  7,  0 ],  //2
  [  0,  7,  0 ],  //3
  [  0,  0,  5 ],  //4
  [ 10,  0,  5 ],  //5
  [ 10,  7,  5 ],  //6
  [  0,  7,  5 ]]; //7

CubeFaces = [
  [0,1,2,3],  // bottom
  [4,5,1,0],  // front
  [7,6,5,4],  // top
  [5,6,2,1],  // right
  [6,7,3,2],  // back
  [7,4,0,3]]; // left

polyhedron( CubePoints, CubeFaces );
```



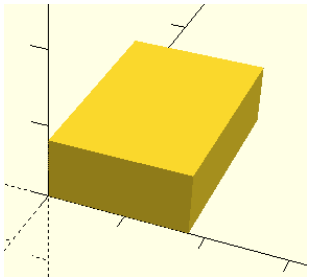
On peux créer des variables

Les transformations

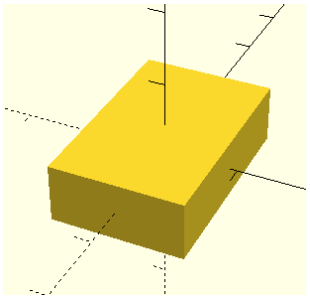
Auteur Marc Forner pour SoFAB

Les transformations

le centrage



cube([18,28,8]);

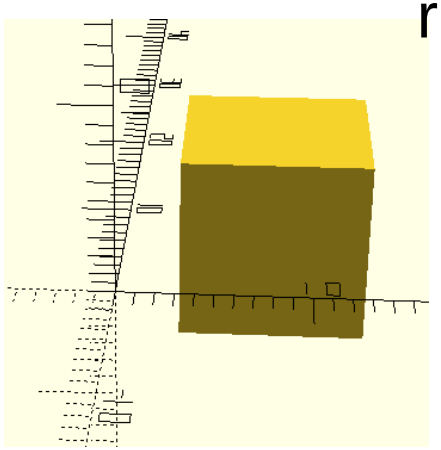


cube([18,28,8],center=true);

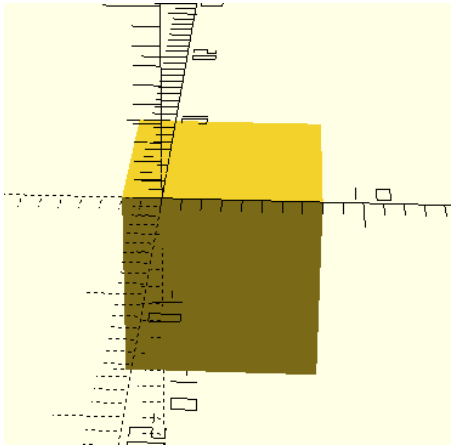
Les transformations

la translation

La translation est en valeur relative, par rapport au point d'origine de l'objet.



translate([3,5,-5]) cube(10);

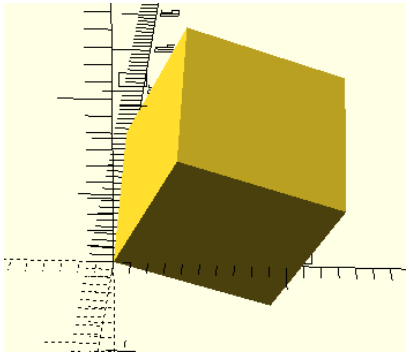


translate([3,5,-5]) cube(10,true);

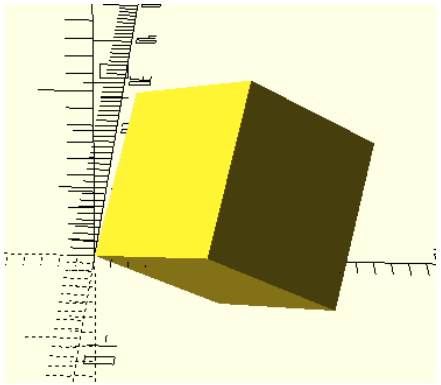
Les transformations

la rotation

La rotation est aussi en valeur relative



rotate([30,20,10]) cube(10);

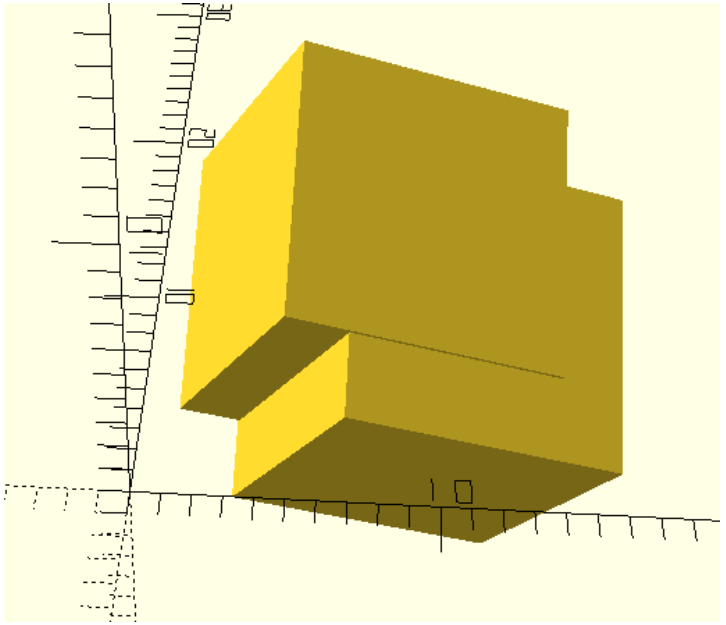


rotate([30,20,10]) rotate([30,20,10]) cube(10);

Les transformations

la rotation

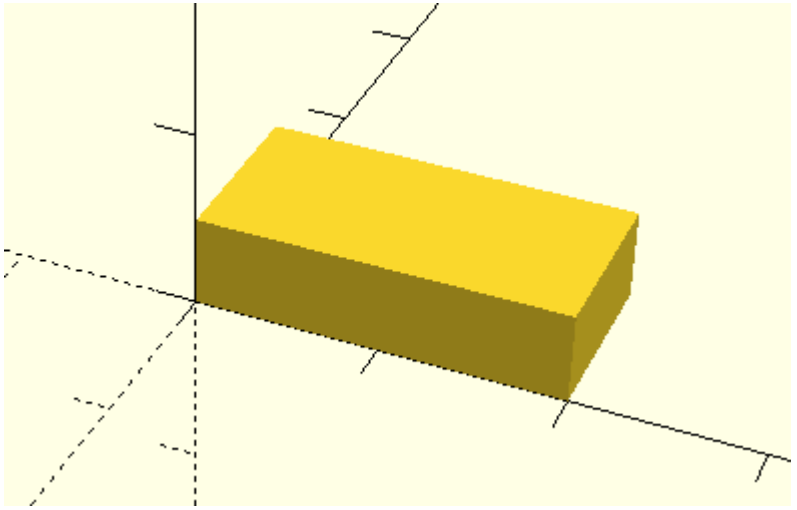
ATTENTION ! translate+rotate ≠ rotate+translate

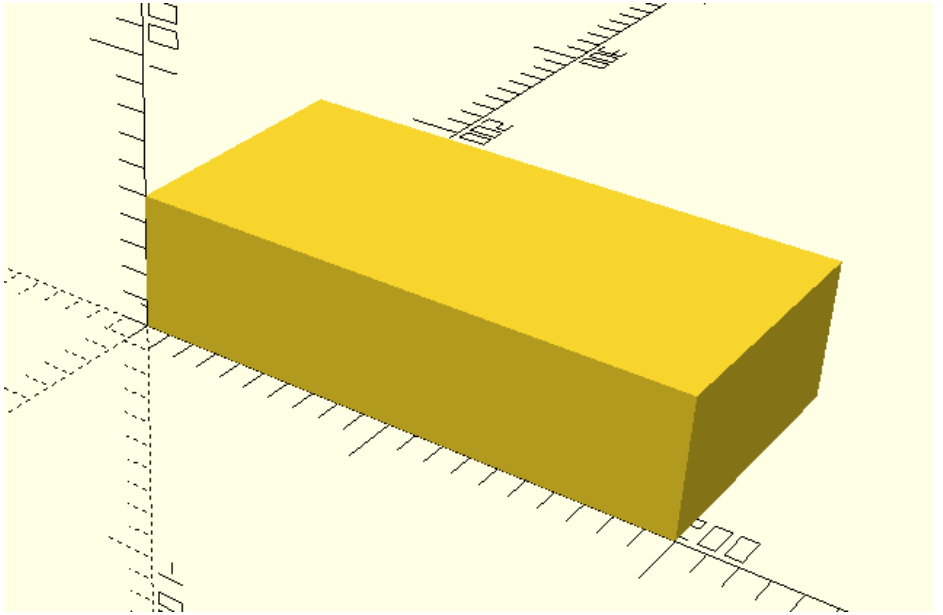


translate([3,5,-5]) rotate([30,20,10]) cube(10);
rotate([30,20,10]) translate([3,5,-5]) cube(10);

Les transformations

redimensionnement



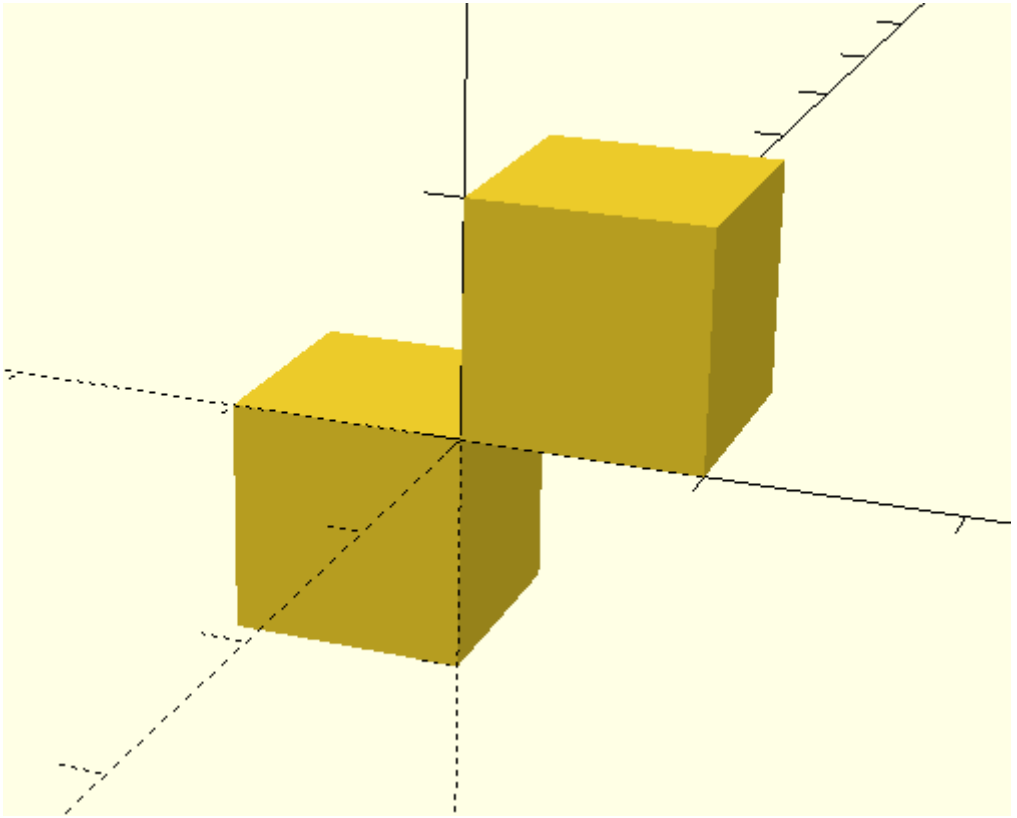


resize([20,10,5]) cube(10);

scale([20,10,5]) cube(10);

Les transformations

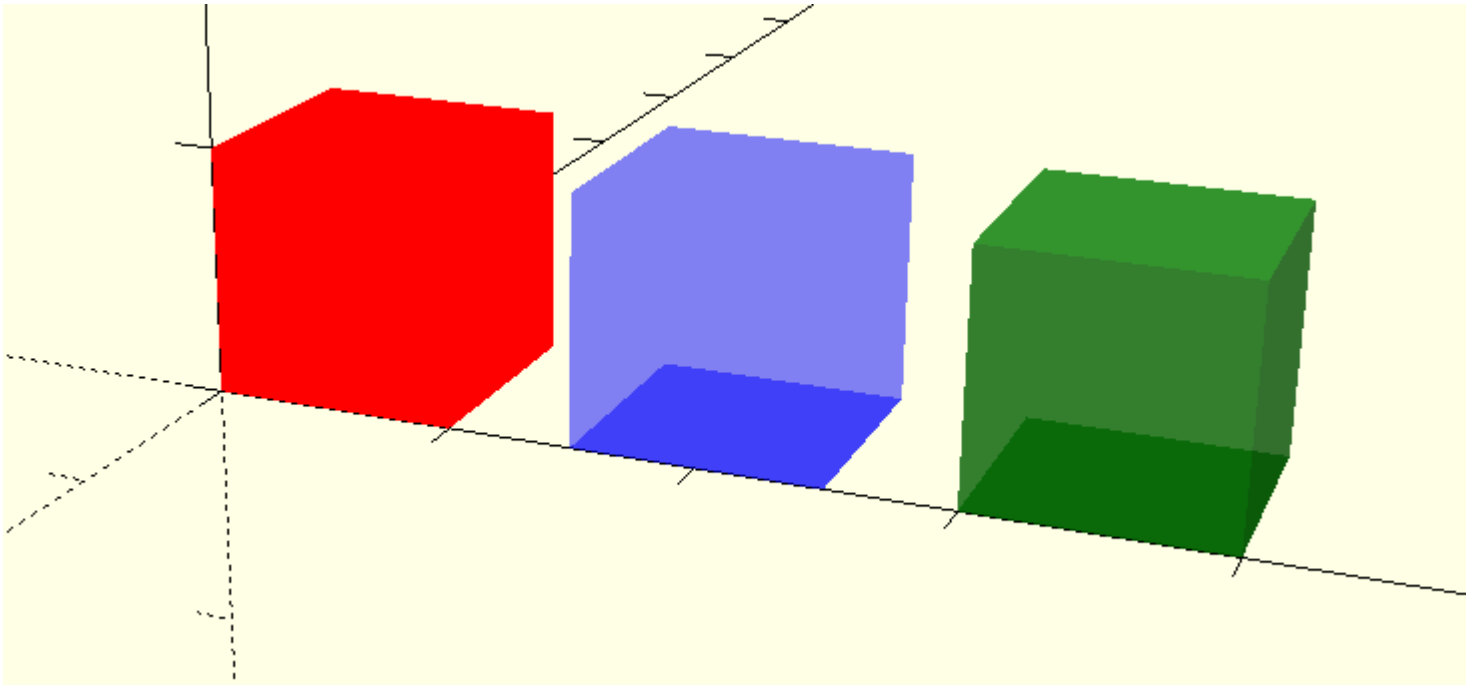
le miroir



```
cube(10);
mirror([1,0,1]) cube(10);
```

Les transformations

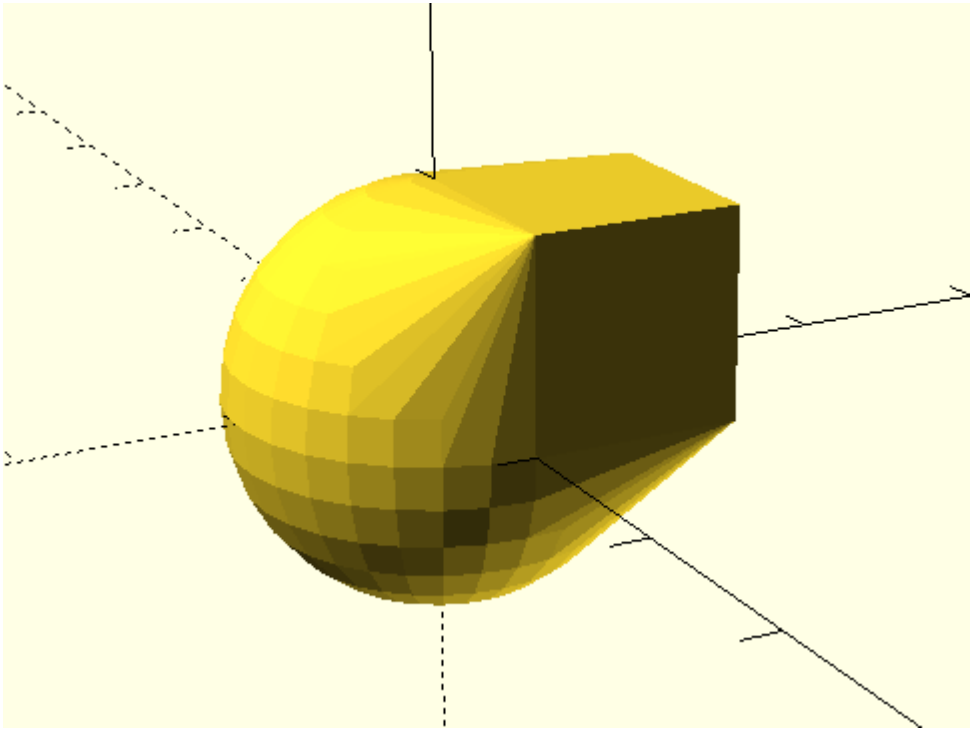
la couleur



```
color([255,0,0]) cube(10);

color([0,0,255,0.5])
translate([15,0,0]) cube(10);

color("green",0.8 )
translate([30,0,0]) cube(10);
```

Les transformations

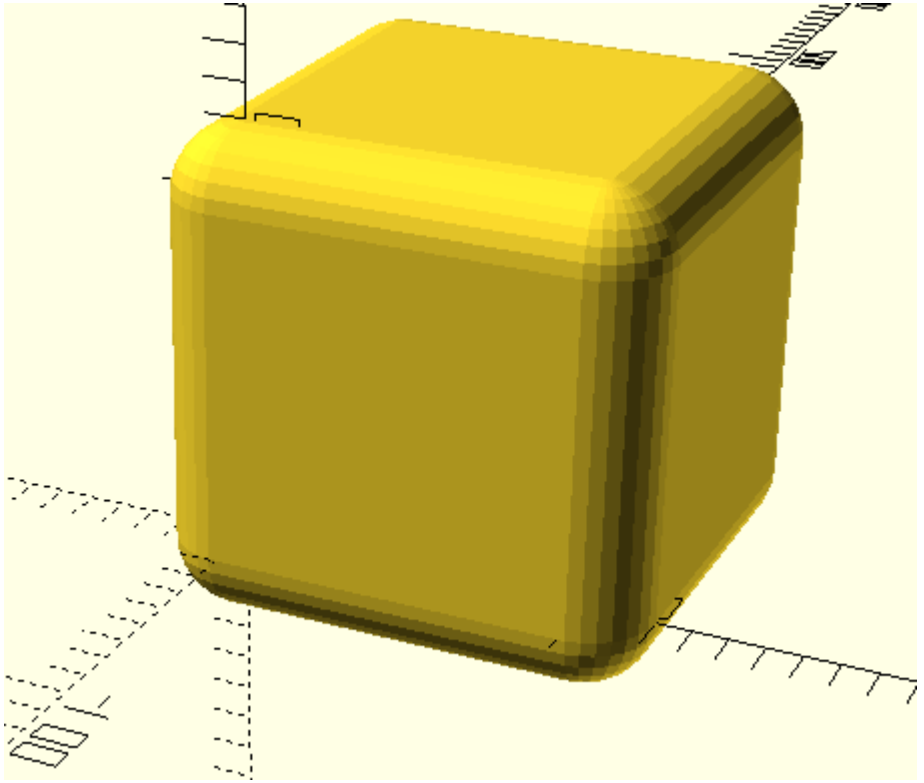
hull



```
hull() {
    cube(10);
    sphere(10);
}
```

Les transformations

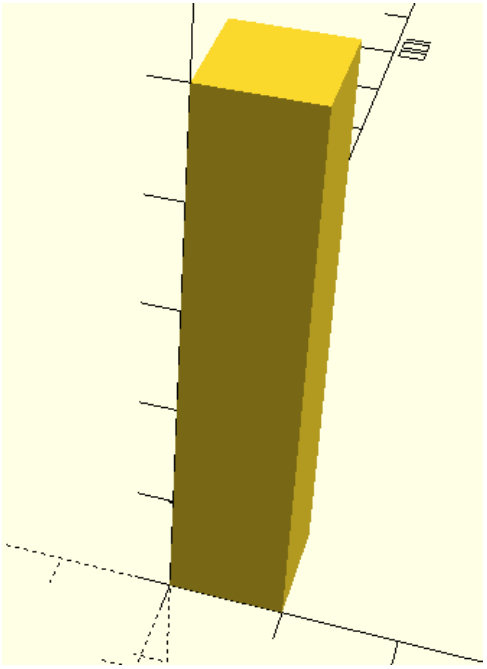
minkowski



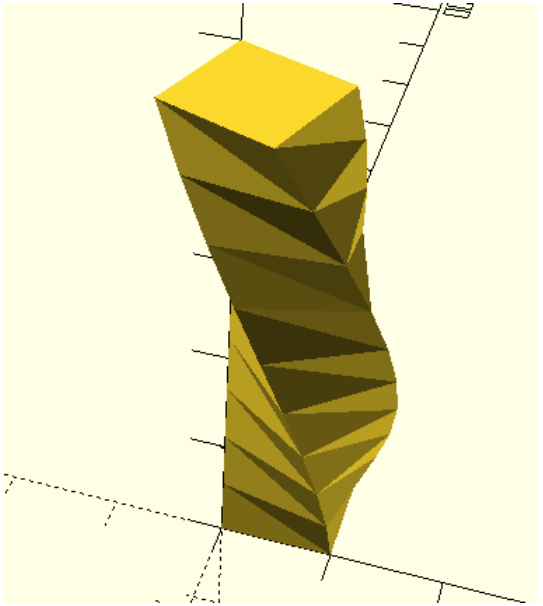
```
minkowski() {
    cube(100);
    sphere(20);
}
```

Les transformations

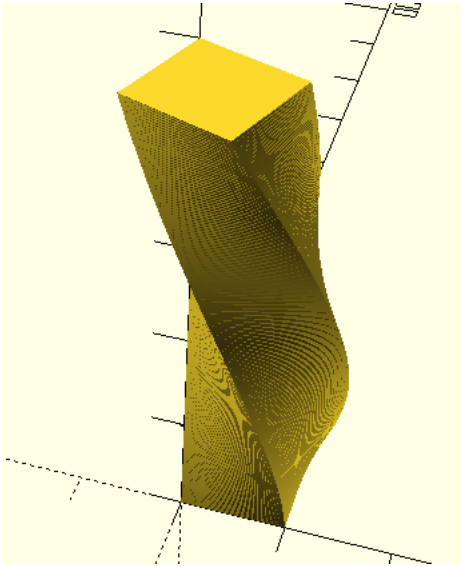
l'extrusion linéaire



linear_extrude
(height=50)
square(10);



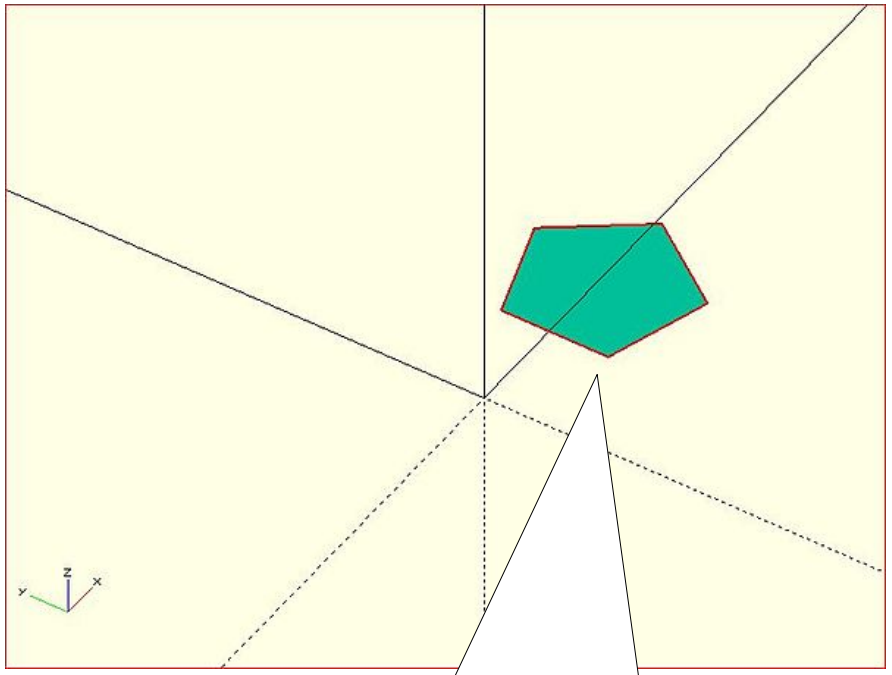
linear_extrude
(height=50, twist=110,)
square(10);

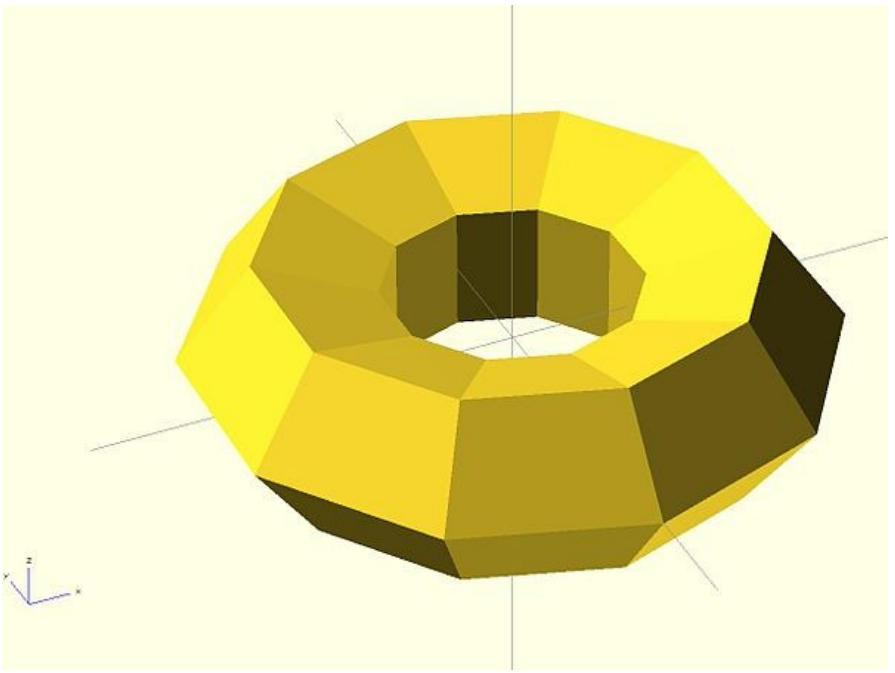


linear_extrude
(height=50, twist=110,
slices=200) square(10);

Les transformations

l'extrusion rotative





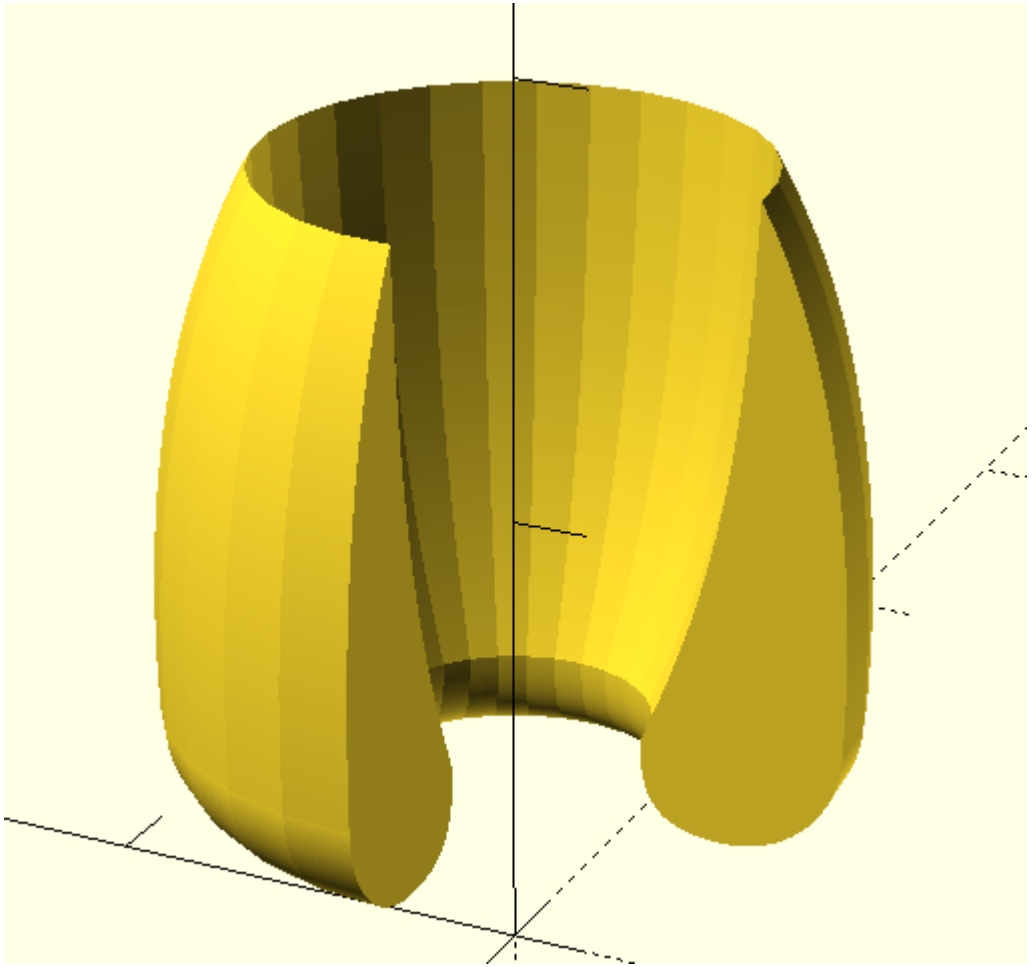
Il existe aussi des primitives 2D :

- Circle (size, center, $fn)
- Square ([x,y])
- Polygon ([points])
- Text (« texte », size, font….)

```
rotate_extrude()
translate([2, 0, 0])
circle(r = 1);
```

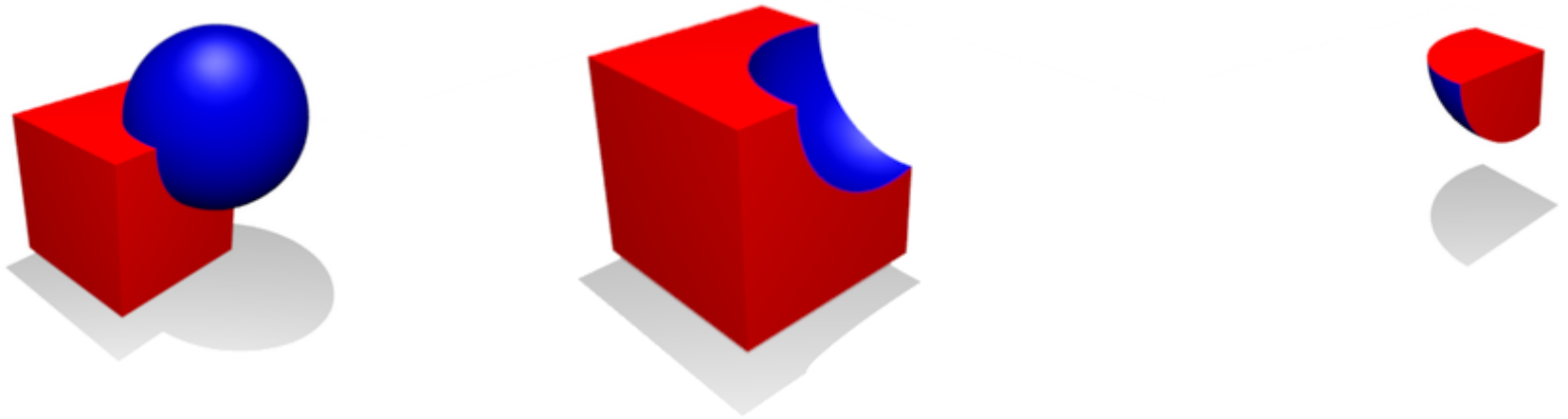
Les transformations

extrusion rotative & l'import de fichiers externes



```
rotate_extrude (angle=270)
rotate([0,0,90]) translate([0,30,0])
import (file="profil.dxf");
```

Les opérations booléennes



Les opérations booléennes

boolléen () {

forme 1

forme 2

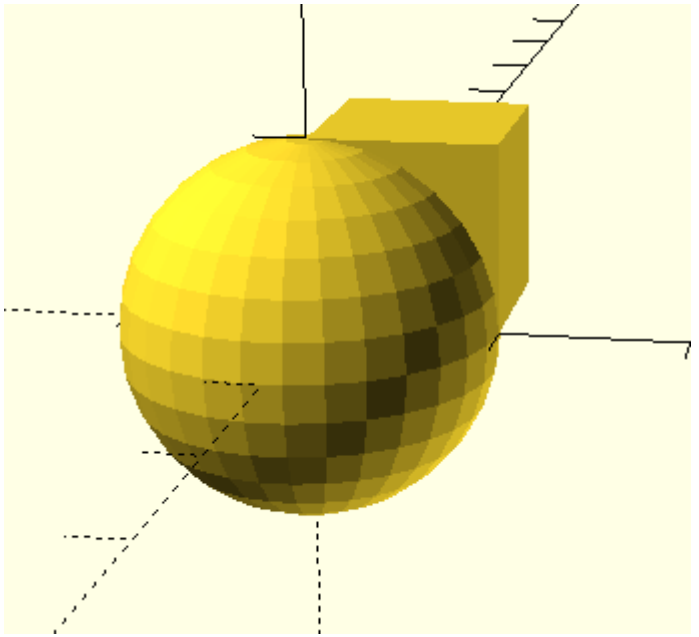
forme 3

....

}

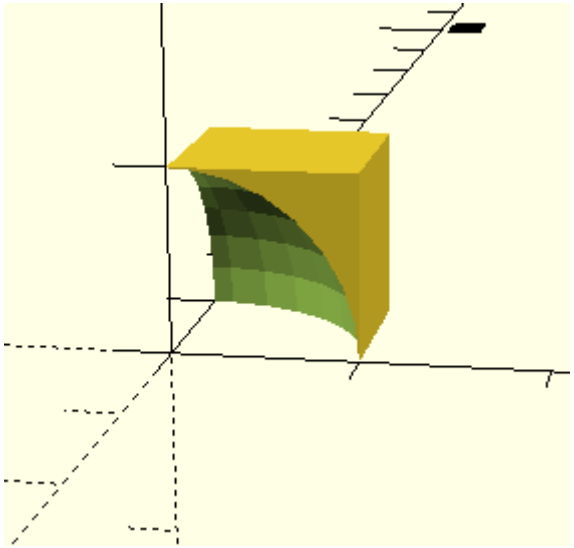
Les opérations booléennes

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



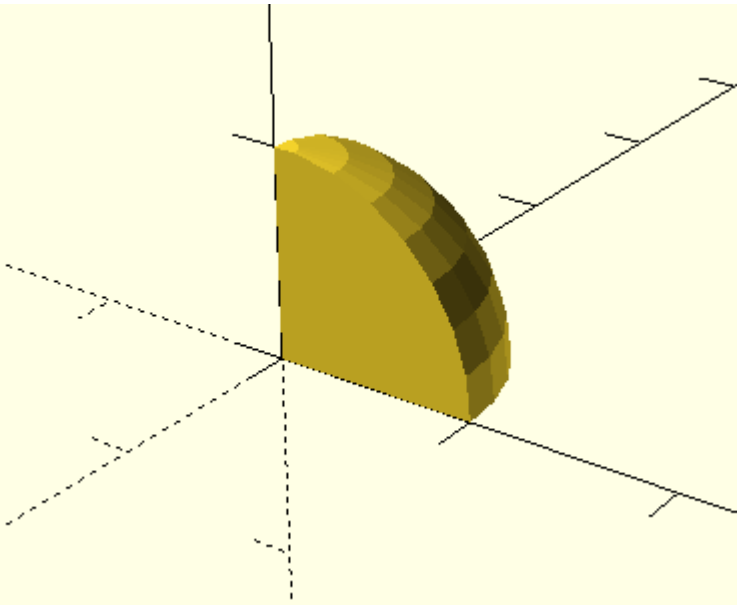
Les opérations booléennes

```
difference () {
    cube(10) ;
    sphere(10) ;
  }
```



Les opérations booléennes

```
intersection () {
    cube(10) ;
    sphere(10) ;
  }
```



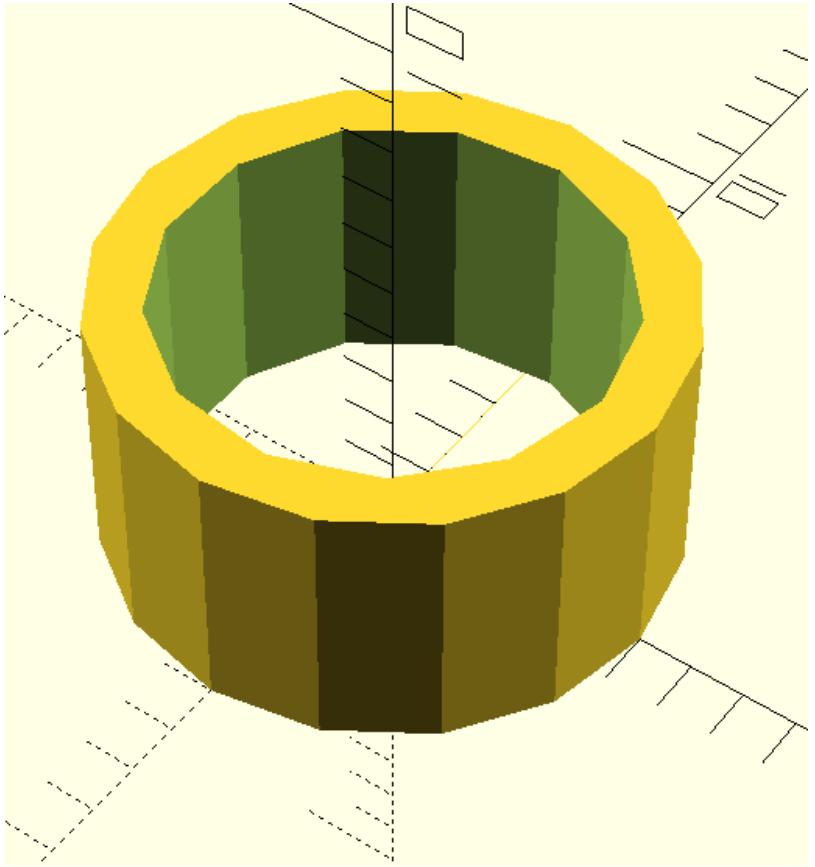
Les modules

Auteur Marc Forner pour SoFAB

Les modules

définir du code réutilisable - exemple 1

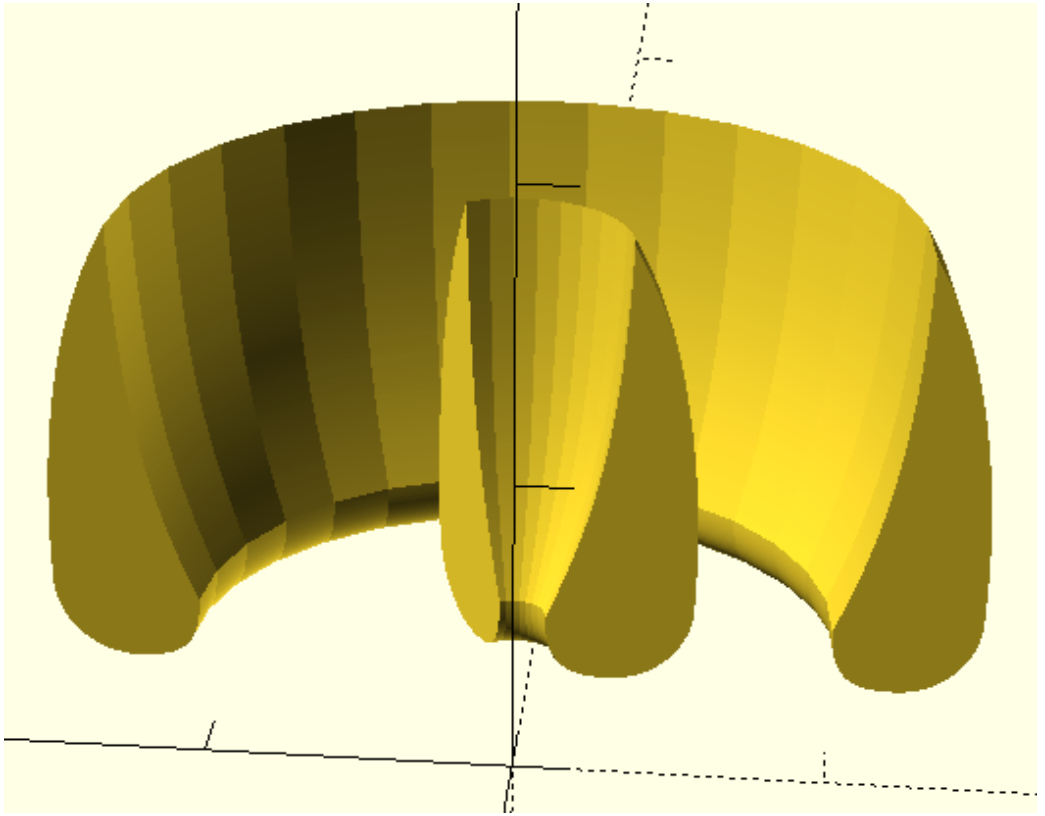
```
module rondelle (di, de, e) {

      difference ()  {
           cylinder (r=de/2, h=e);
           translate ([0,0,-1]) cylinder (r=di/2,h=e+2);
     }
}

rondelle(8,10,5);
```



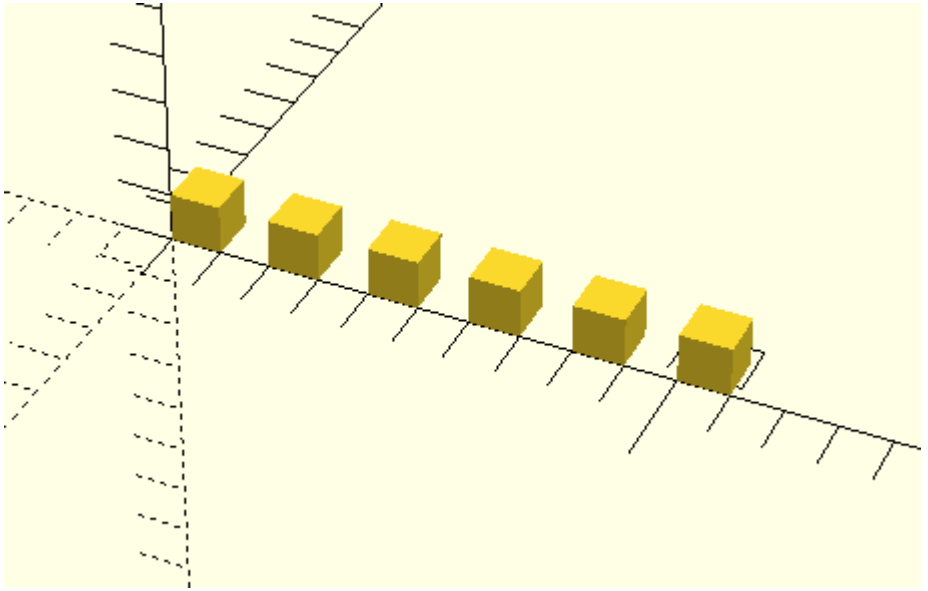
Les modules

définir du code réutilisable - exemple 2

```
module vase(angle,diametre) {

    rotate_extrude(angle=angle)
    rotate([0,0,90]) translate([0,diametre,0])
    import (file="profil.dxf");
}

vase(180,100);
vase(120,10);
```



L’itération

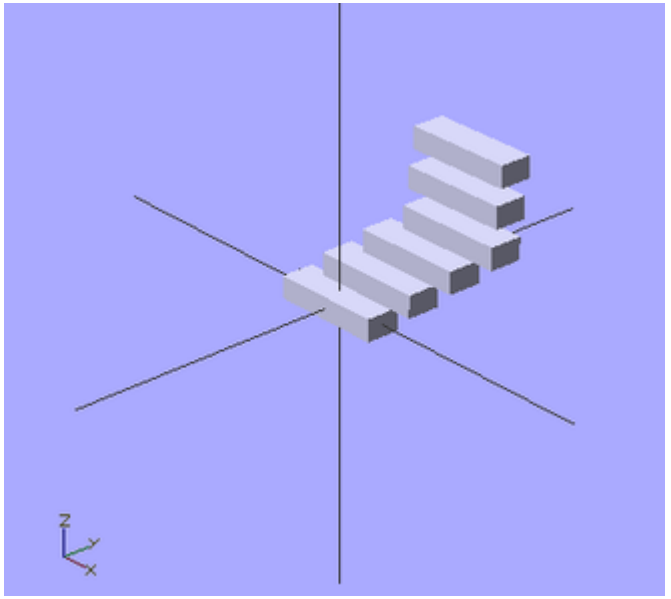


```
for (i=[0:5]) translate([2*i,0,0]) cube(1);
```

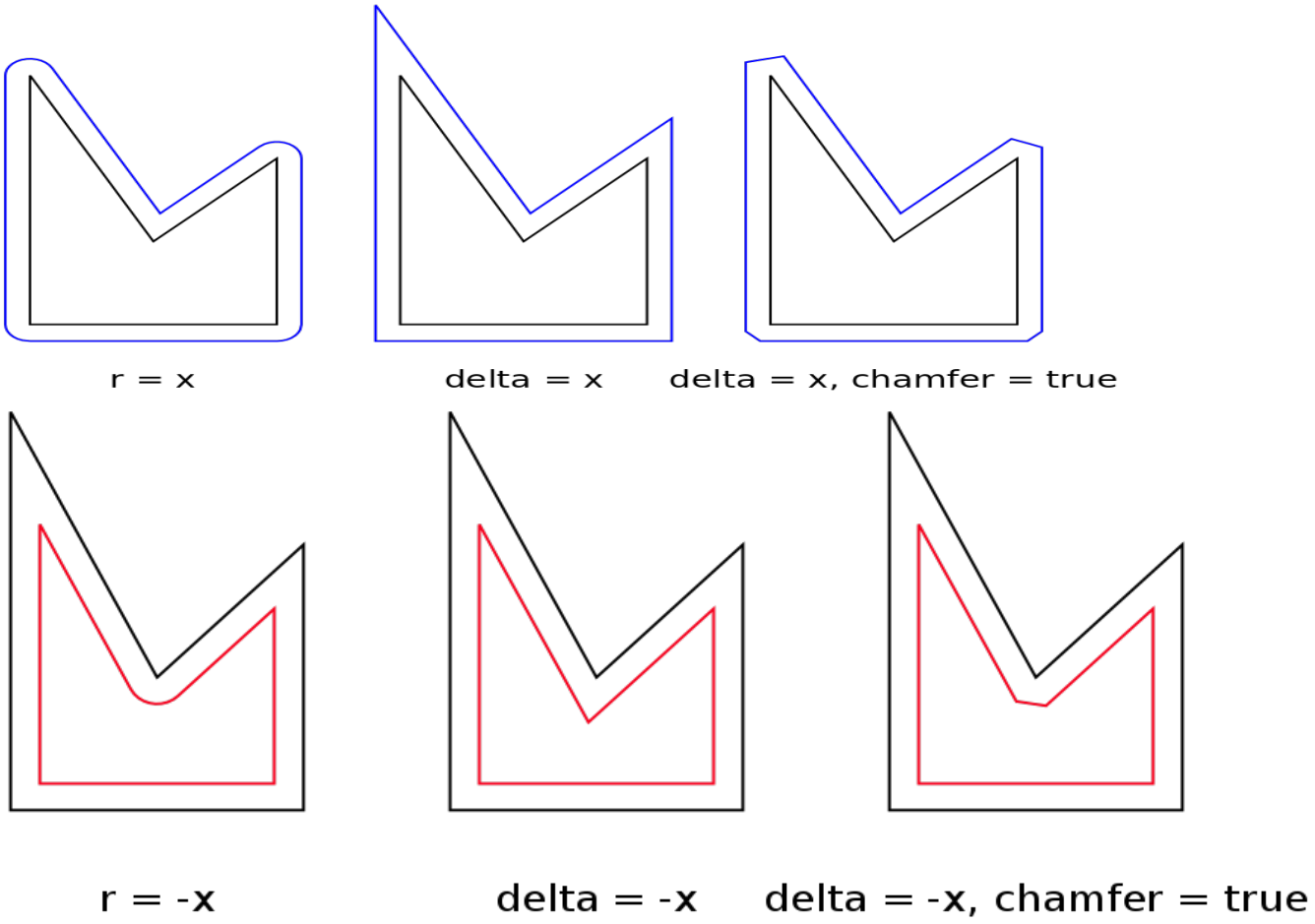
L’itération

utilisation des matrices

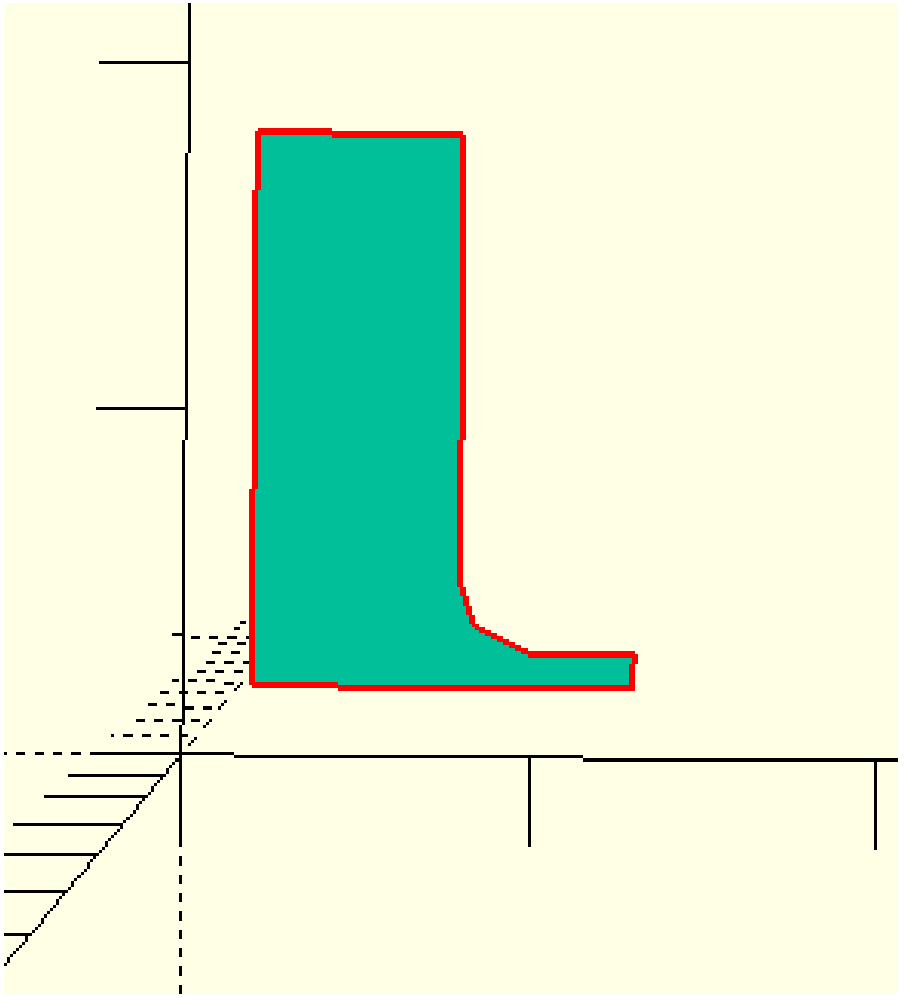
```
for(i = [ [ 0,  0,  0],
          [10, 12, 10],
          [20, 24, 20],
          [30, 36, 30],
          [20, 48, 40],
          [10, 60, 50] ])
{
   translate(i)
   cube([50, 15, 10],
center = true);
}
```



Chamfrein et congé

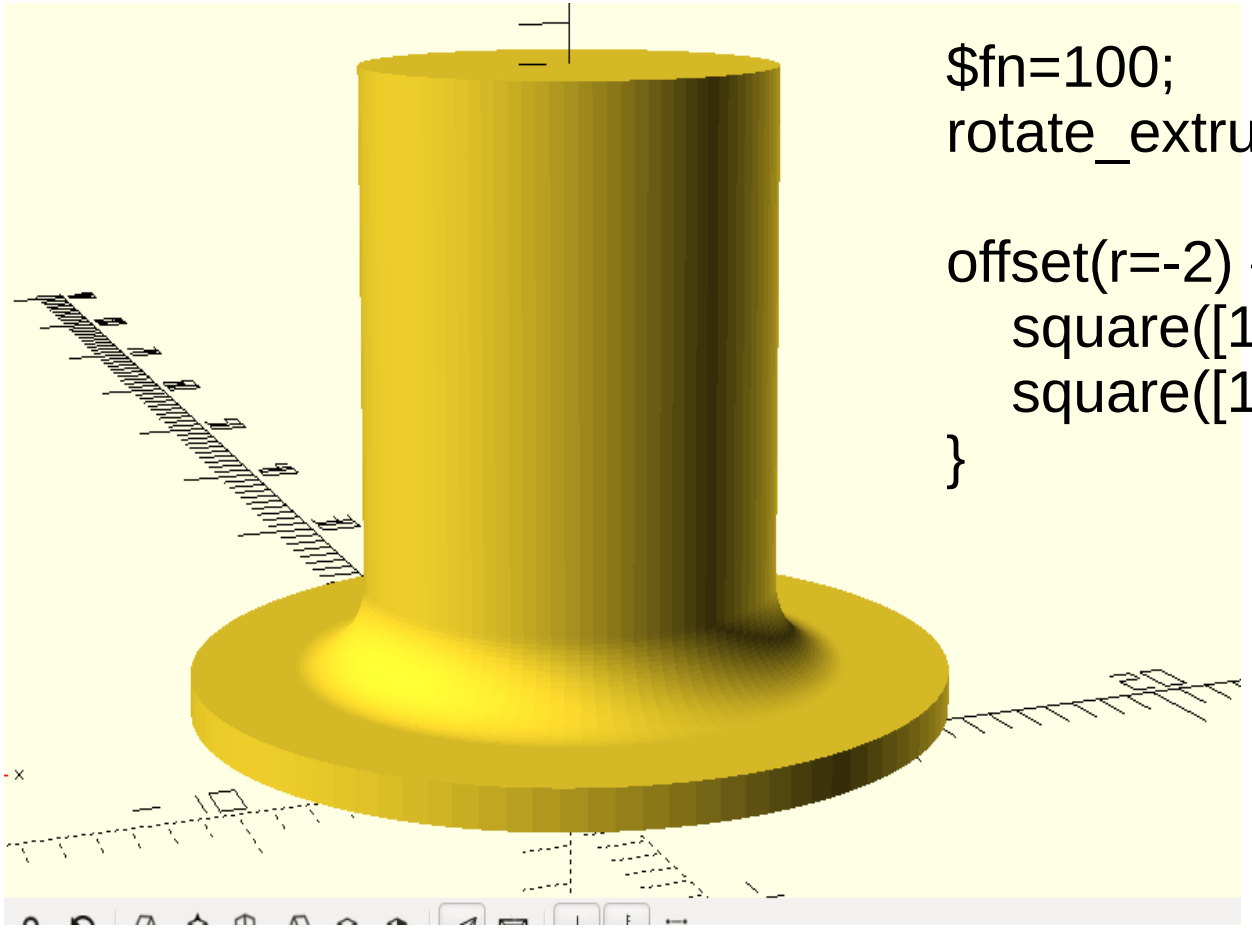


Chamfrein et congé



```
offset(r=-2) {
    square([15,5]);
    square([10,20]);
}
```

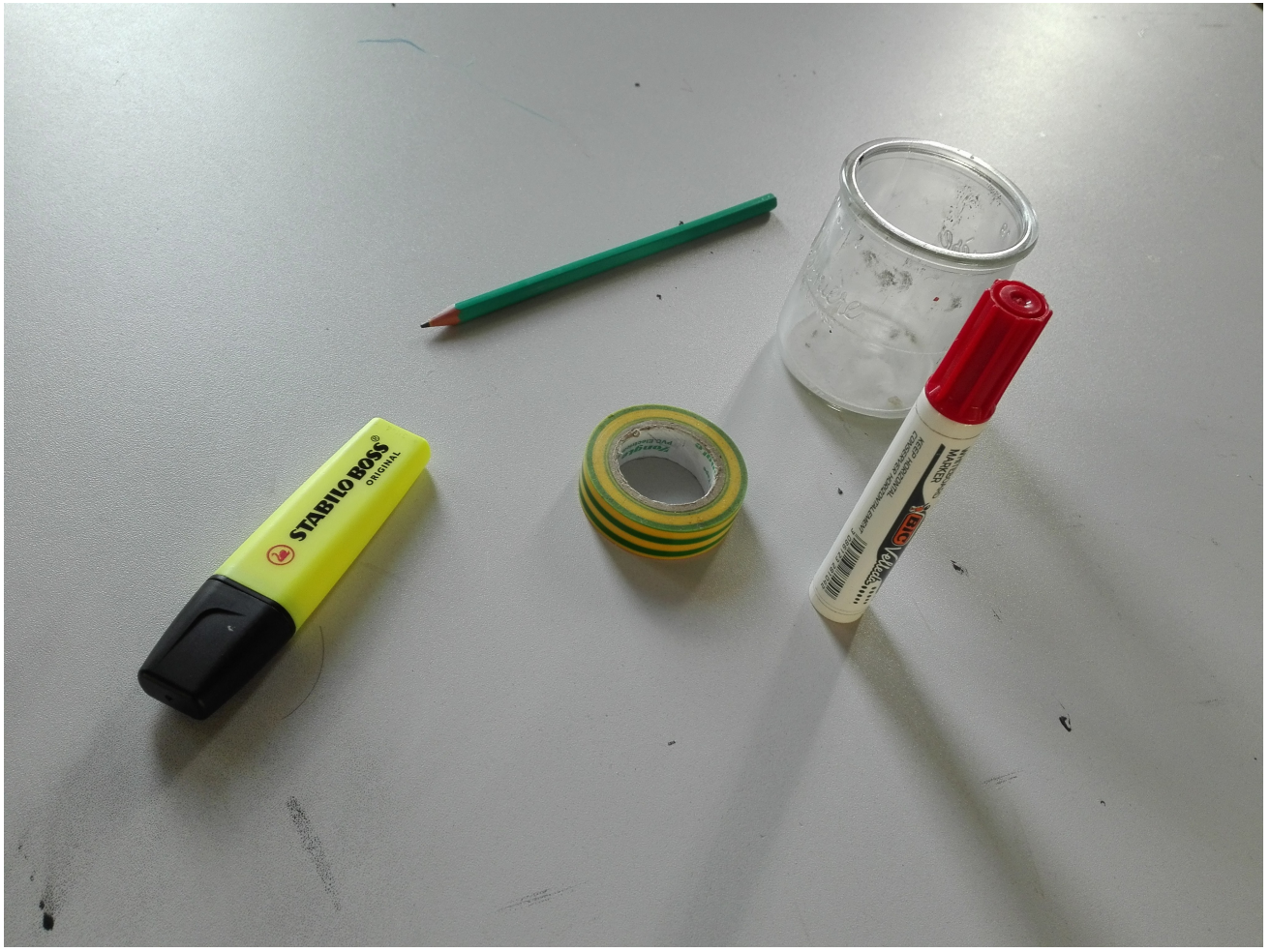
Chamfrein et congé



```
$fn=100;
rotate_extrude() translate([-2,0,0])

offset(r=-2) {
    square([15,5]);
    square([10,20]);
}
```

Conseil

Pour débuter sans se démotiver, commencez par reproduire des objets courants simples



Conseil

plan

briques

rendu

```
VARIABLES
```

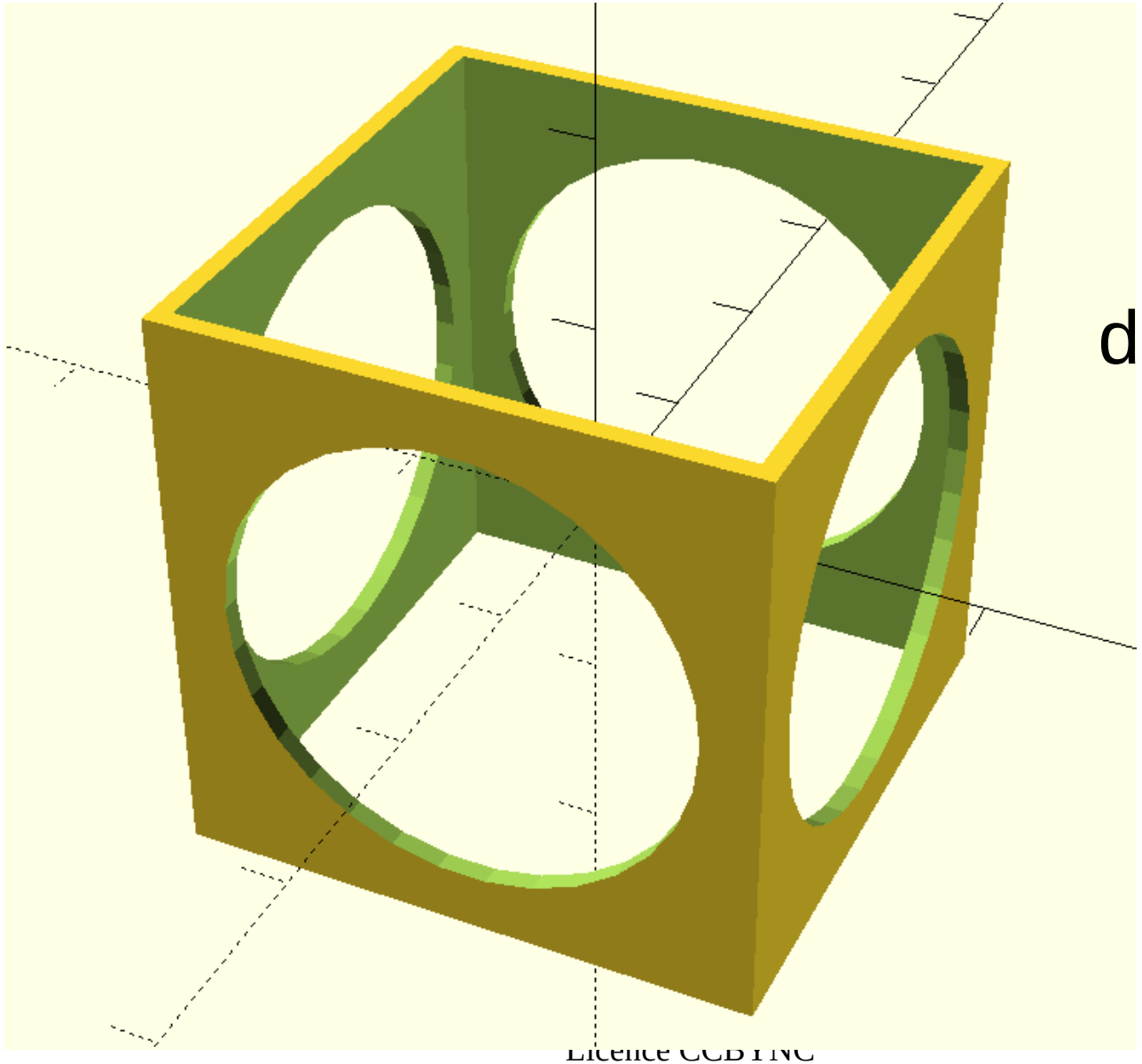
```
module x () {
        boolléen() {
                primitives1
                primitives2
}
```

```
difference() {

        group() {
                primitives1
                Module x
                …..
        }

primitive2
module y
…..
}
```

matière

matière à enlever

Exercice 1

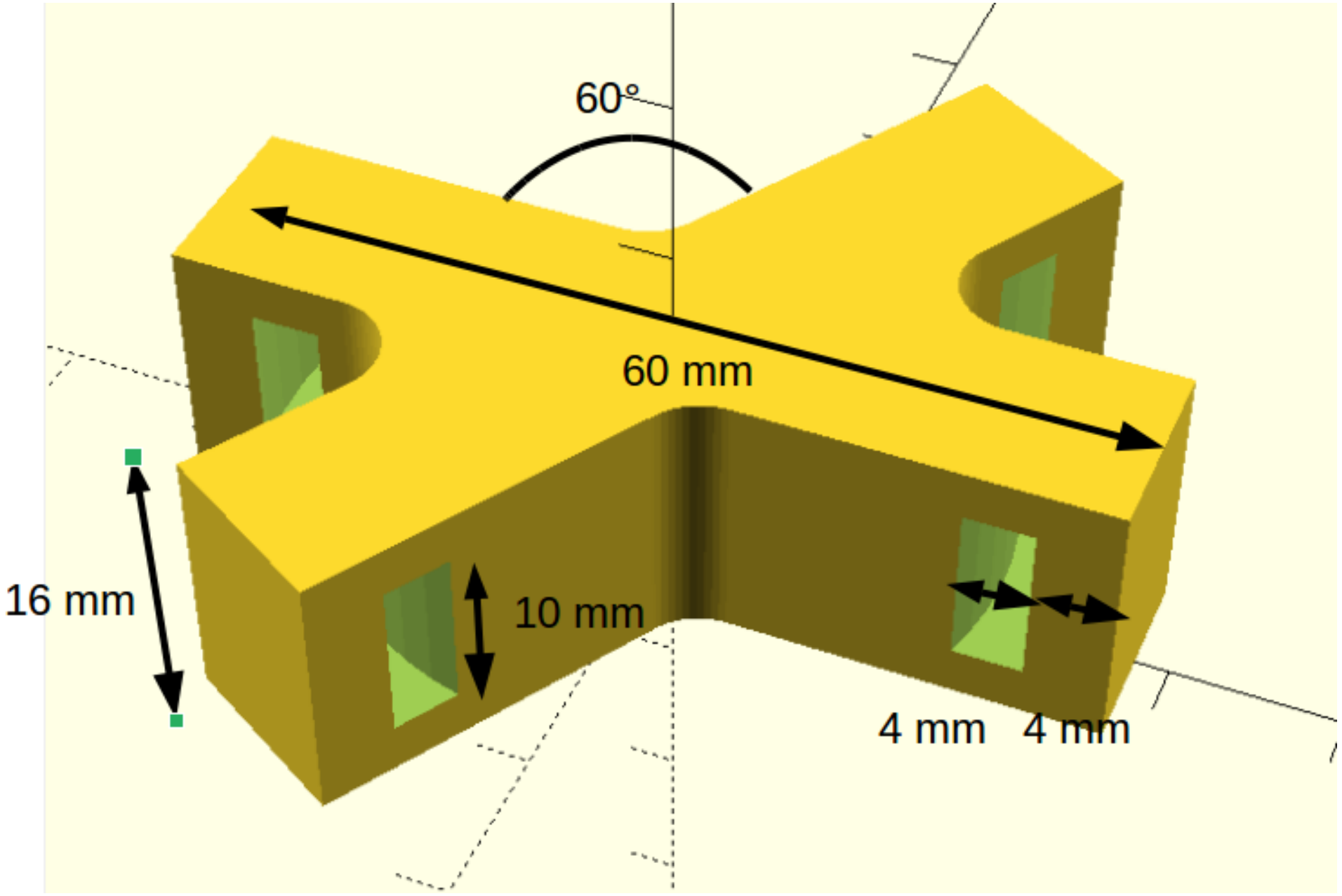


Cube 30 mm
diamètre trou 24mm
épaisseur 1mm

Exercice 1

```
difference() {
    cube(30,true);
cube([28,28,31],true);
rotate([90,0,90])
cylinder(r=12,h=60,center=true);
    rotate([90,0,0])
cylinder(r=12,h=60,center=true);

}
```

Exercice 2



```
$fn=60;

module forme() {

    offset(r=-4) {
       rotate([0,0,60])  square([64,20],true);
       square([64,20],true);
    }
}

difference() {

    linear_extrude (height=16) forme();
    translate([0,0,3]) rotate_extrude() translate([20,0])
square([4,10]) ;

}
```