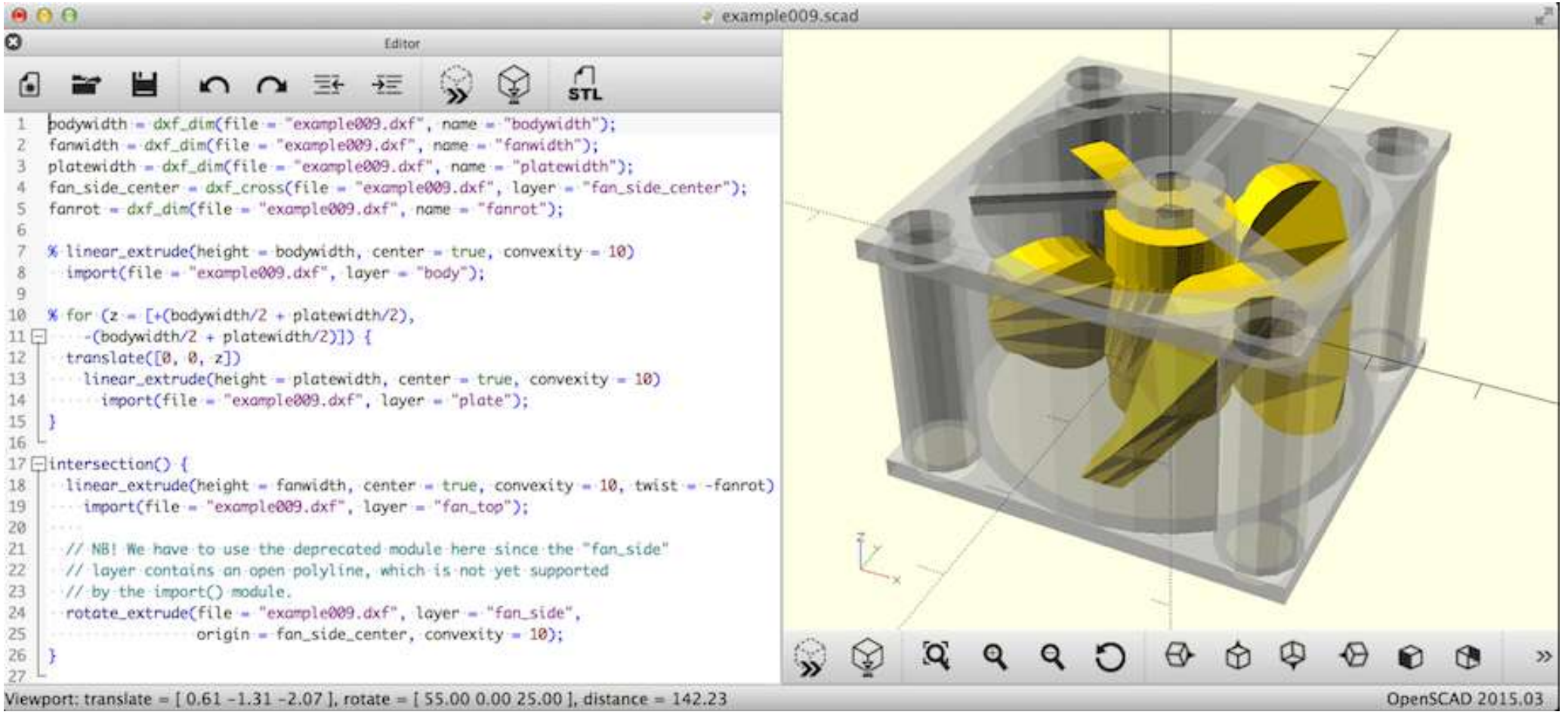
OpenSCAD

CAD Paramétrique



R. Ziegler Mars 2016

Dessin en trois étapes

Dans le menu Conception, trois commandes correspondent aux trois étapes du dessin d'un objet, dont la dernière est l'exportation au format STL utilisé en impression 3D par les logiciels d'écriture du Gcode :

- affichage du dessin en zone de visualisation, par le menu Conception > Aperçu ou touche F5. C'est une commande répétée à chaque modification du code, pour en voir l'effet,
- rendu final, par Conception > Rendu ou touche F6,
- enregistrement en STL, par le menu Fichier > Exporter > Exporter comme STL…

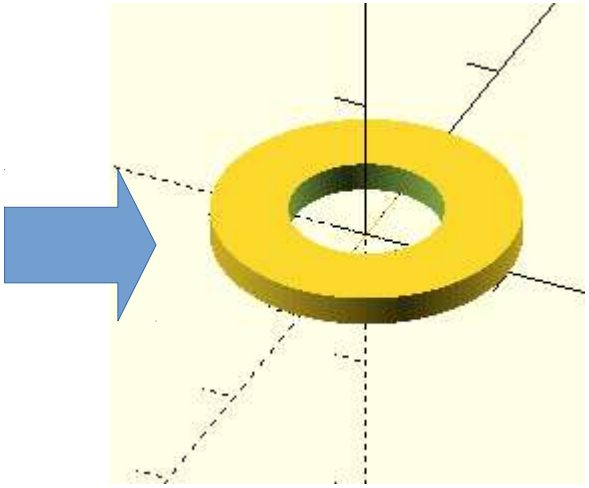
Ces trois commandes ont des raccourcis dans l'interface utilisateur. L'exportation en STL doit être faite après le rendu (F6), rien ne se passe si elle est demandée avant. D'autres commandes accessibles par les menus ne sont pas détaillées ici.

Souris dans la zone de visualisation : le bouton gauche permet de pivoter le dessin, le droit de le déplacer, la roue centrale d'en modifier la taille d'affichage.

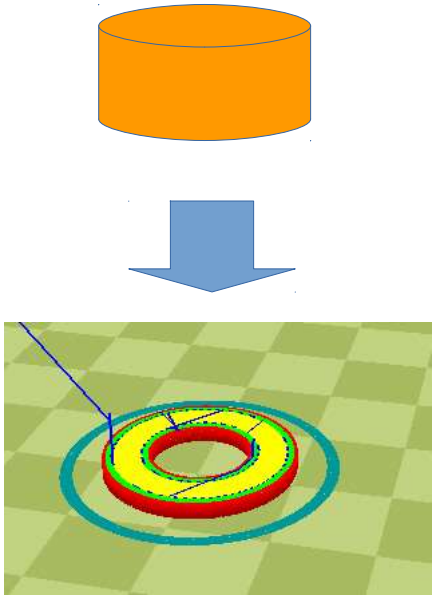
Dessin en trois étapes


```
/*
  paramètres :
  di : diamètre intérieur
  de : diamètre extérieur
  e  : épaisseur
*/
rondelle(10,20,2);

module rondelle (di, de, e)
{
  $fn=100;
  difference ()
  {
    cylinder (r=de/2, h=e);
    translate ([0,0,-1]) cylinder
(r=di/2, h=e+2);
  }
}
```

Aperçu 3d touche F5
Rendu 3d touche F6



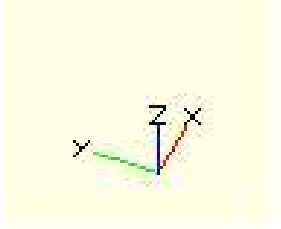
Ficher Stereo-lito
STL pour imprimante
3d



Axes X, Y, Z et 1er essai

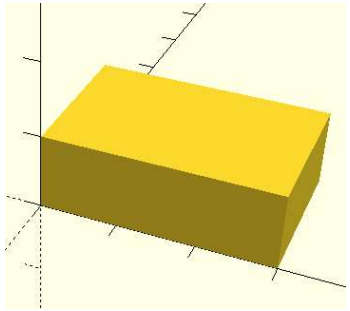
Axes X, Y et Z

Les trois axes du dessin sont X pour la largeur, Y pour la profondeur et Z pour la hauteur. Dans le menu Vue. Afficher les Axes permet d'afficher ces axes.



1er essai

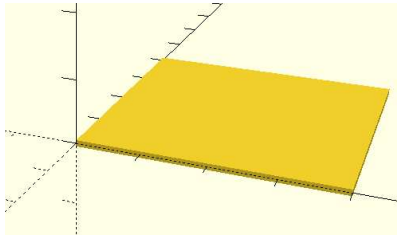
Saisir cube ([30,20,10]); dans la zone d'édition puis appuyer la touche F5. Un parallélépipède est alors dessiné. Le code signifie « dessine-moi un "cube" de 30 mm de large, 20 de profondeur et 10 de hauteur ».



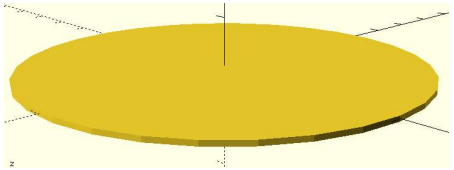
Commandes de dessin 2D

Trois commandes dessinent des formes 2D :

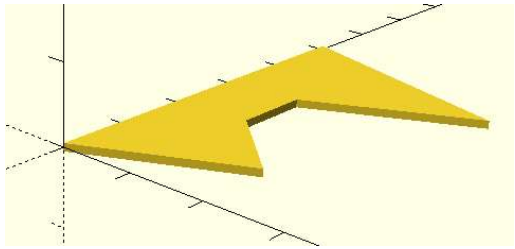
square() pour un carré ou rectangle,
circle() pour un cercle,
polygon() pour un polygone.



square(10,20);



circle(10);



polygon(points=[[0,0], [20,10],[10,20],[10,30], [30,40],[0,50]]);

Elles n’ont pas d’épaisseur, celle-ci leur est donnée par les commandes de transformation linear_extrude() et rotate_extrude().

Commandes de dessin 3D

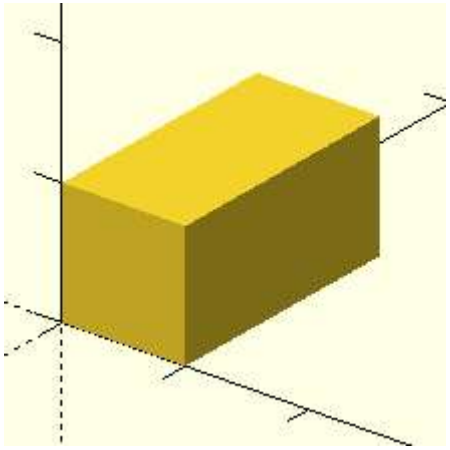
OpenSCAD permet de créer quatre formes 3D simples avec les commandes suivantes :

cube() pour un parallélépipède,

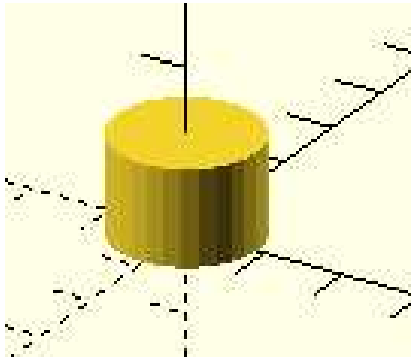
cylinder() pour un cylindre ou un cône, tronqué ou non,

sphere() pour une sphère,

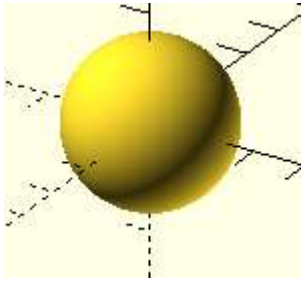
polyhedron() pour un polyèdre.



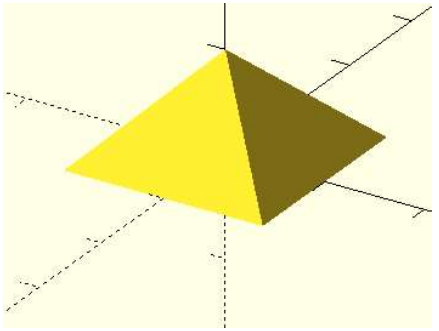
cube([10,50,20]);



cylinder (r1=9, r2=9, h=12);



sphere(d=30);



polyhedron(points =[...],faces[…]) ;

Commandes de transformation

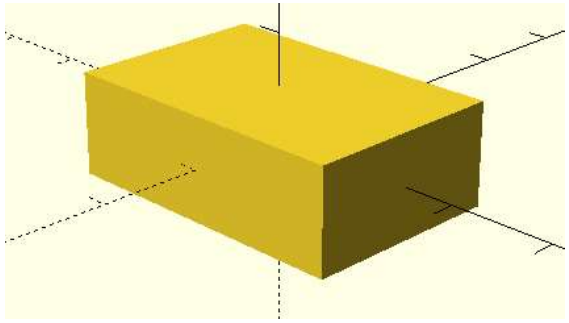
Les formes 3D et 2D peuvent être combinées et manipulées pour former des objets complexes, par des commandes de transformation dont voici les principales :

- **difference()** pour extraire une forme d'une autre
- **union()** pour unir plusieurs éléments
- **translate()** pour déplacer un élément
- **rotate()** pour pivoter un élément
- **intersection()** pour ne conserver que la partie commune de deux formes se chevauchant
- **resize()** pour modifier la taille d'un élément
- **color()** pour colorer l'affichage d'un élément
- **hull()** pour réaliser une fusion de formes
- **minkowski()** pour créer un enveloppement d'une forme par une autre,
- **import()** pour utiliser un fichier STL ou DXF,
- **linear_extrude()** et **rotate_extrude()** pour donner une élévation à une forme 2D

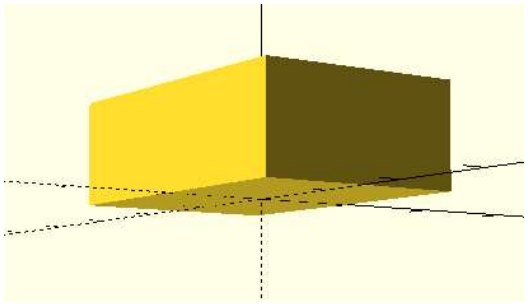
Par exemple, translate ([0,0,5]) cube ([30,20,10], center=true);

Exemple de transformation

cube ([30,20,10], center=true);

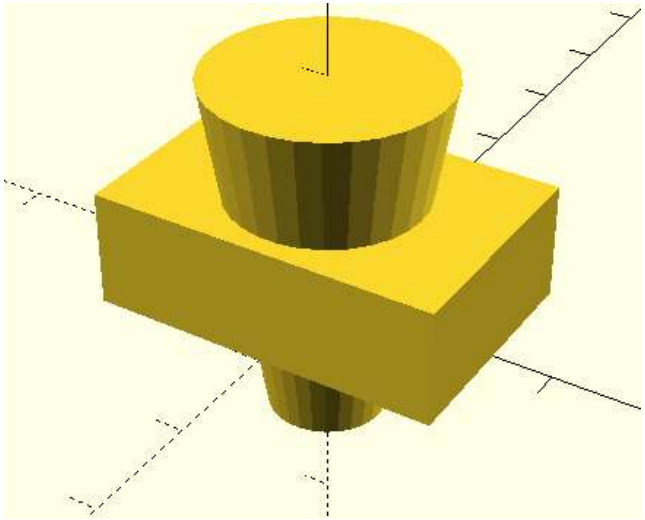


translate ([0,0,5]) cube ([30,20,10], center=true);

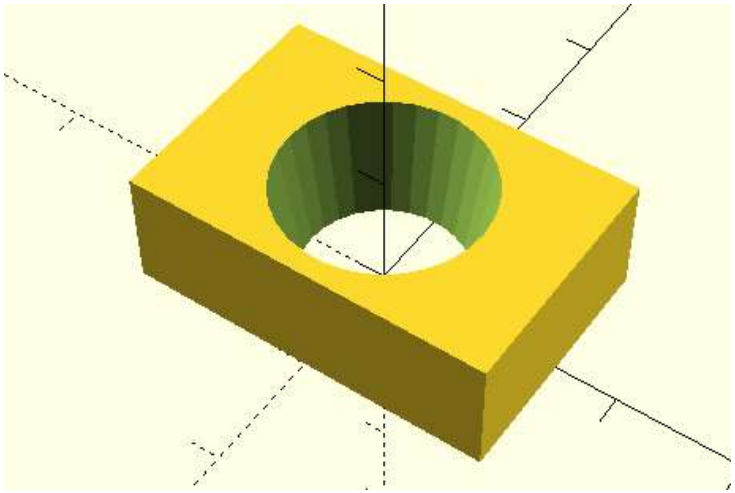


Union and Difference

```
union(){
translate ([0,0,5]) cube ([30,20,10],
center=true);

translate ([0,0,-10]) cylinder
(r=5,r2=10,h=30);

}
```

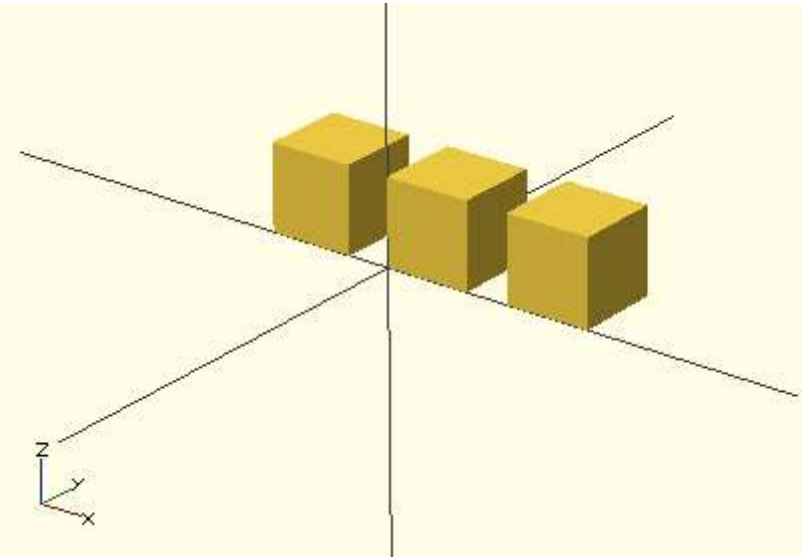


```
difference(){
translate ([0,0,5]) cube ([30,20,10],
center=true);

translate ([0,0,-10]) cylinder
(r=5,r2=10,h=30);

}
```



Utilisation de variables

```
a = 10;

cube (a);

// on ajoute 5 à la valeur de a, pour le déplacement en X vers la droite
translate ([a+5, 0, 0]) cube (a);

// on soustrait 5 à la valeur de -a, pour le déplacement en X vers la gauche
translate ([-a-5, 0, 0]) cube (a);
```



combinaison

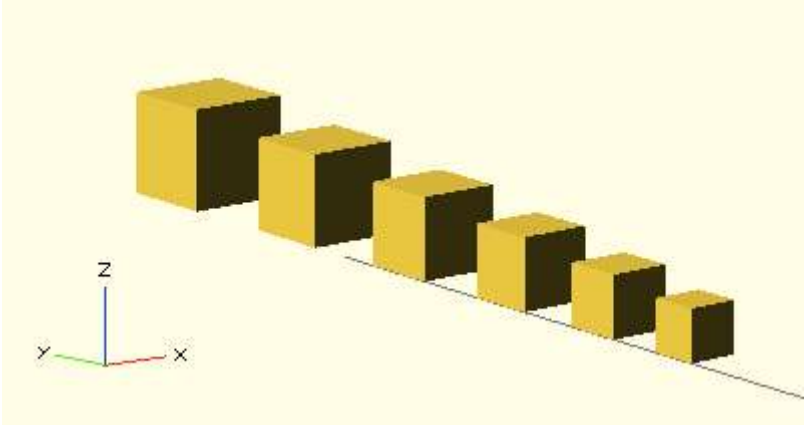
```
// example cylindre percé
h=10;
r=3;
s=1;
hole=2;
circle_resolution=40;

test() ;
```

```
module test(){

    difference(){ // trou dans cylindre creux
        difference(){ // cylindre creux
            cylinder(      h = h,
                    r = r,
                    center = true,
                    $fn = circle_resolution);
            cylinder(      h = h+1,
                    r = r-s/2,
                    center = true,
                    $fn = circle_resolution);
             }
             translate([0,h/2,0])
             rotate([90,0,0])
             cylinder(
                  h=h, // len
                  r=hole, // diam
                  $fn = circle_resolution);
        }
    }
```

Itérations et conditions

On peut répéter une commande plusieurs fois avec une boucle for, ou l'exécuter sous condition if / else. Voici la page du manuel. Pour dessiner six cubes de tailles et d'espacements progressifs :

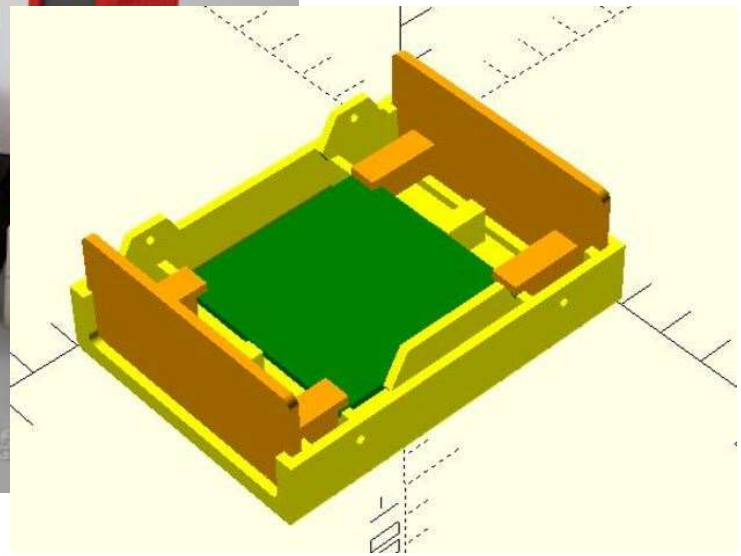
```
for (i = [5:10])  // pour i valant de 5 à 10
{
  translate ([0, i*i, 0]) cube (i);  // dessin d'un cube d'arête i
}
```



Exemple1

http://www.thingiverse.com/thing:1367661

Exemple 2



http://www.thingiverse.com/thing:1333774

Exemple 3

https://www.thingiverse.com/thing:941081

https://www.thingiverse.com/thing:941081

Aide mémoire

Syntax

```
var = value;
module name(…) { … }
name();
function name(…) = …
name();
include <….scad>
use <….scad>
```

2D

```
circle(radius)
square(size,center)
square([width,height],center)
polygon([points])
polygon([points],[paths])
```

3D

```
sphere(radius)
cube(size)
cube([width,height,depth])
cylinder(h,r,center)
cylinder(h,r1,r2,center)
polyhedron(points, triangles, convexity)
```

Transformations

```
translate([x,y,z])
rotate([x,y,z])
scale([x,y,z])
mirror([x,y,z])
multmatrix(m)
color("colorname")
color([r, g, b, a])
hull()
minkowski()
```

Boolean operations

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

Modifier Characters

```
*   disable
!   show only
#   highlight
%   transparent
```

Mathematical

```
abs
sign
acos
asin
atan
atan2
sin
cos
floor
round
ceil
ln
len
log
lookup
min
max
pow
sqrt
exp
rands
```

Other

```
echo(…)
str(…)
for (i = [start:end]) { … }
for (i = [start:step:end]) { … }
for (i = […,…,…]) { … }
intersection_for(i = [start:end]) { … }
intersection_for(i = [start:step:end]) { … }
intersection_for(i = […,…,…]) { … }
if (…) { … }
assign (…) { … }
search(…)
import("….stl")
linear_extrude(height,center,convexity,twist,slices)
rotate_extrude(convexity)
surface(file = "….dat",center,convexity)
projection(cut)
render(convexity)
```

Special variables

```
$fa minimum angle
$fs minimum size
$fn number of fragments
$t  animation step
```