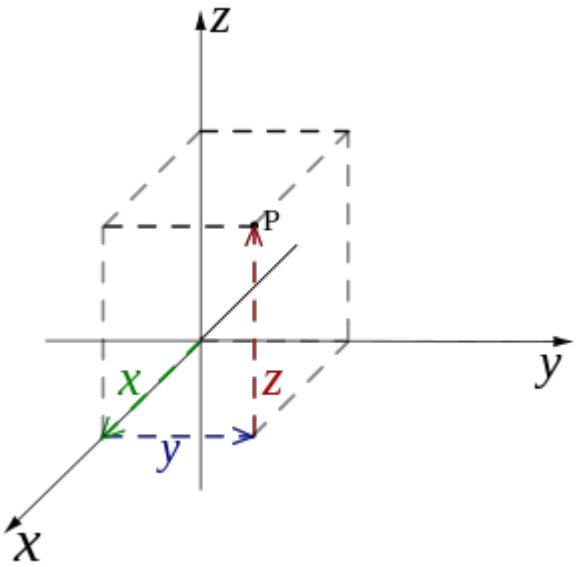
Programmation de volumes avec OpenSCAD

OpenSCAD est un logiciel libre disponible pour Windows, MacOS et Linux.
L'adresse pour télécharger de logiciel est : http://www.openscad.org/downloads.html

Ce logiciel permet de programmer avec un langage qui lui est propre des volumes les plus divers.

La base de fonctions (instructions) de OpenSCAD ce sont les coodonnées spatiales X, Y et Z.

X = largeur
Y = profondeur
Z = hauteur



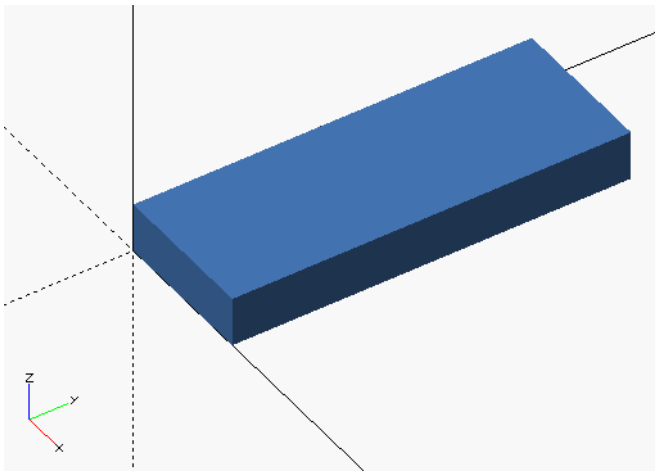
La première fonction à connaître s'écrit :

```
cube ([ X, Y, Z ]) ;
```

elle finit obligatoirement par un point-virgule.

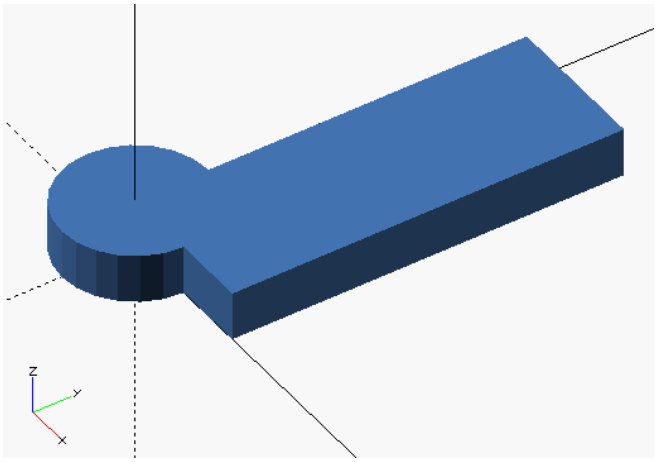
Un premier programme

```
cube([15,40,5]);
```



Continuons notre programme

Nous allons inclure une « variable ».
Une variable est un « mot » de notre choix qui contient la valeur d'un nombre.



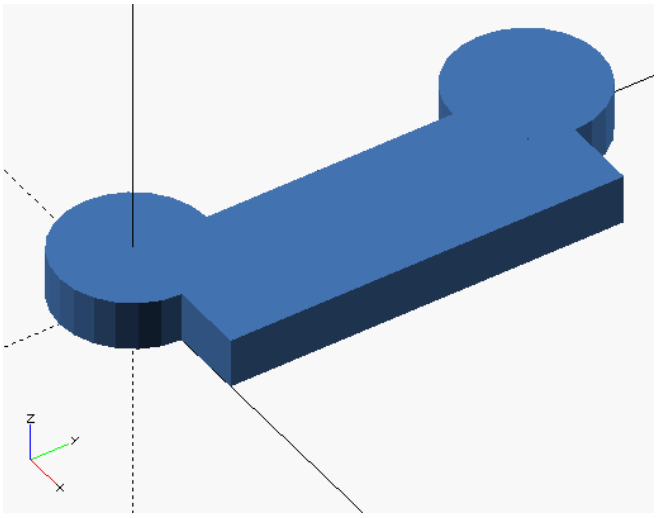
Les variables facilitent d'applique une valeur à différents endroits de notre programme et éventuellement faire des calculs

Puis nous allons ajouter un cylindre dont la rayon est la moitié de la largeur (X) de notre cube :

```
larg = 15;
cube([larg,40,5]);
cylinder(r=larg/2,h=15);
```

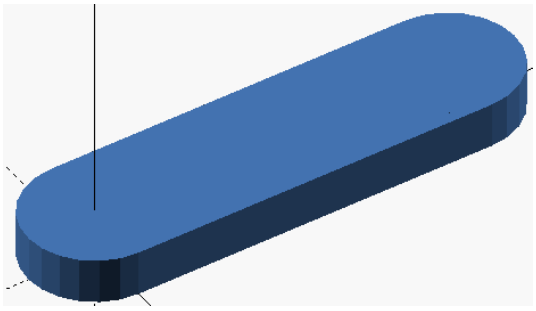
Le cylindre s'écrit par une fonction qui contient :
r = rayon du cylindre (r = larg/2) et
h = hauteur du cylindre

Maintenant nous voulons ajouter un autre cylindre à l'extrémité de notre cube et la intervient la fonction translate (déplacer) toujours sur la base de nos coordonnées X-Y-Z pour déplacer notre deuxième cylindre à l'autre bout du cube :



```
larg = 15;
long = 40;
haut = 5;

cube([larg,long,haut]);
cylinder(r=larg/2,h=haut);

translate([0,long,0])
cylinder(r=larg/2,h=haut);
```

```
larg = 15;
long = 40;
haut = 5;

// déplacement du cube
translate([-larg/2,0,0])
cube([larg,long,haut]);
cylinder(r=larg/2,h=haut);

// déplacement du 2ème cylindre
translate([0,long,0])
cylinder(r=larg/2,h=haut);
```



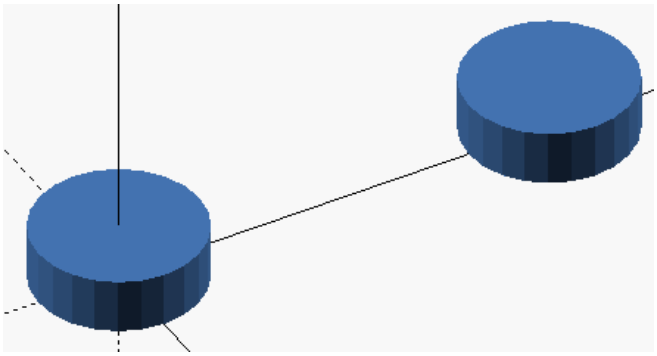
Les deux slash « // » devant un texte quelconque servent à « commenter » des endroits de notre programme ce que est très utile dans des programmes plus longs et complexes.

Allons u peu plus loin…

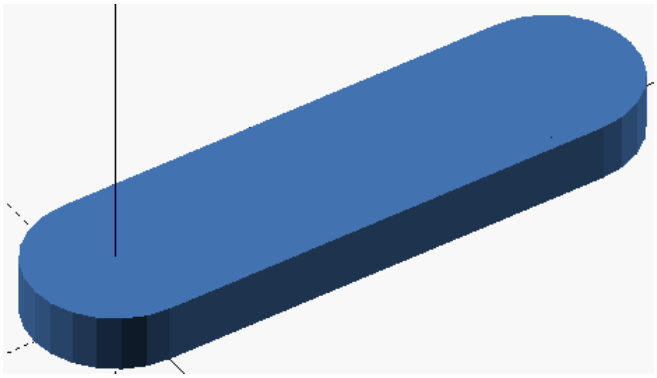
Il y a parfois différentes façon d'arriver à un même résultat en programmation,

Voici nos deux cylindres :

```
larg = 15;
long = 40;
haut = 5;

cylinder(r=larg/2,h=haut);

translate([0,long,0])
cylinder(r=larg/2,h=haut);
```



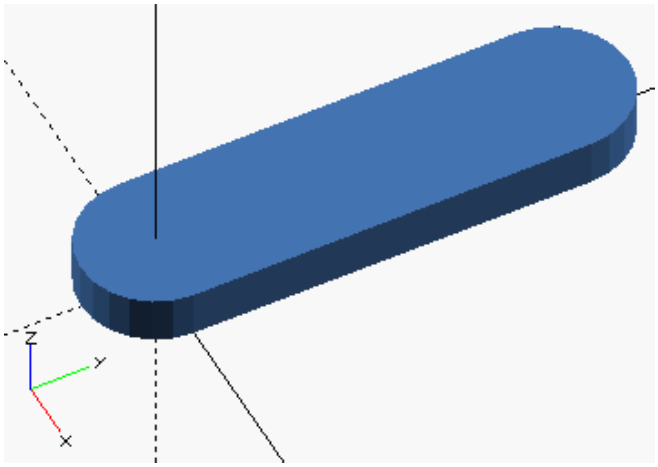
Il existe une fonction appelée **hull()** qui réunit différentes formes qui sont encerclés par des accolades « { ,,, } »



```
larg = 15;
long = 40;
haut = 5;

hull(){
cylinder(r=larg/2,h=haut);

translate([0,long,0])
cylinder(r=larg/2,h=haut);
}
```

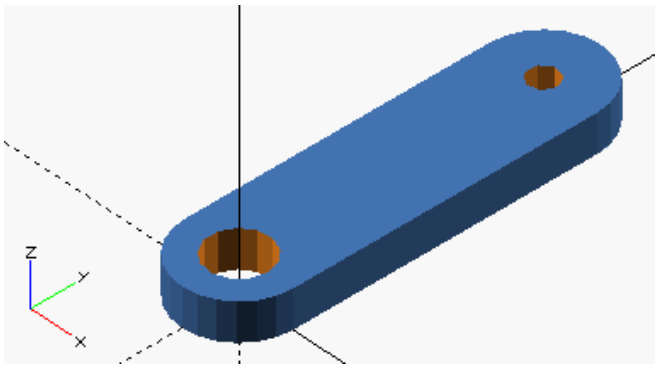
On a pu constater que la fonction translate([X,Y,Z]) ne termine pas par un point-virgule, cela est du à que translate fait partie de l'instruction **cube()** ou l'instruction **cylinder().**

Pour traiter notre programme par sections distinctes noue allons le découper en modules, qui transforment nos scripts en »fonctions » réutilisables.



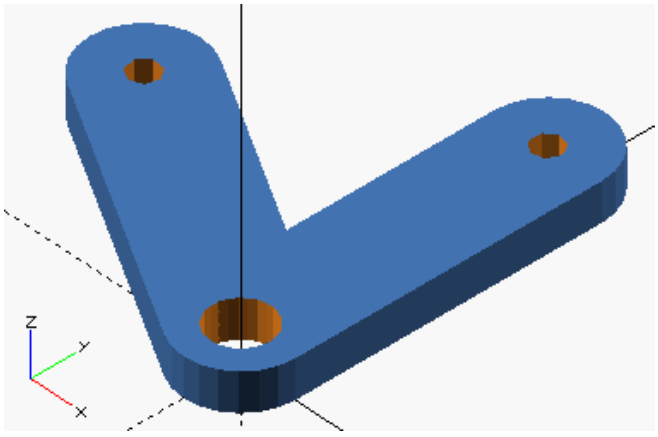
```
larg = 15;
long = 40;
haut = 5;

module barre (){
    hull(){
cylinder(r=larg/2,h=haut);
translate([0,long,0])
cylinder(r=larg/2,h=haut);
    }}

barre(); // afficher le module
```

Un nouvelle fonction **difference()** qui permet de creuser de trous dans notre forme,



```
larg = 15;
long = 40;
haut = 5;

module barre (){
    hull(){
cylinder(r=larg/2,h=haut);

translate([0,long,0])
cylinder(r=larg/2,h=haut);
    }}

module perforations (){
    perf = 2;
difference (){
barre();
translate([0,0,-1])
cylinder(r=perf*2,h=haut+2);
translate([0,long,-1])
cylinder(r=perf,h=haut+2);
} }

perforations();
```

Et encore une fonction : **rotate()**
elle fera tourner la première instruction **perforations()** qu'elle rencontre.
Après le point-virgule la deuxième barre ne sera pas altérée.



```
rotate([0,0,60])
perforations();
perforations();
```