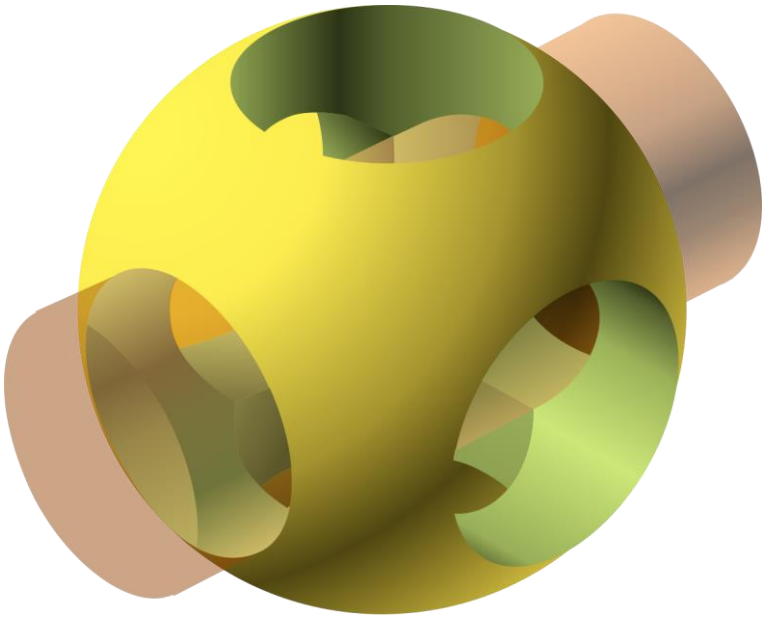
TUTORIEL openSCAD



Réalisé par :
HADDADI Hichem
élève-ingénieur en GE4a à Polytech Clermont

Introduction

OpenSCAD est un logiciel libre, sous licence GPLv.2, de modélisation paramétrique fonctionnant sous Linux, MacOs et Microsoft Windows.

Les objets sur openSCAD sont créés à partir d'un script qui va être exécuter dans une fenêtre.

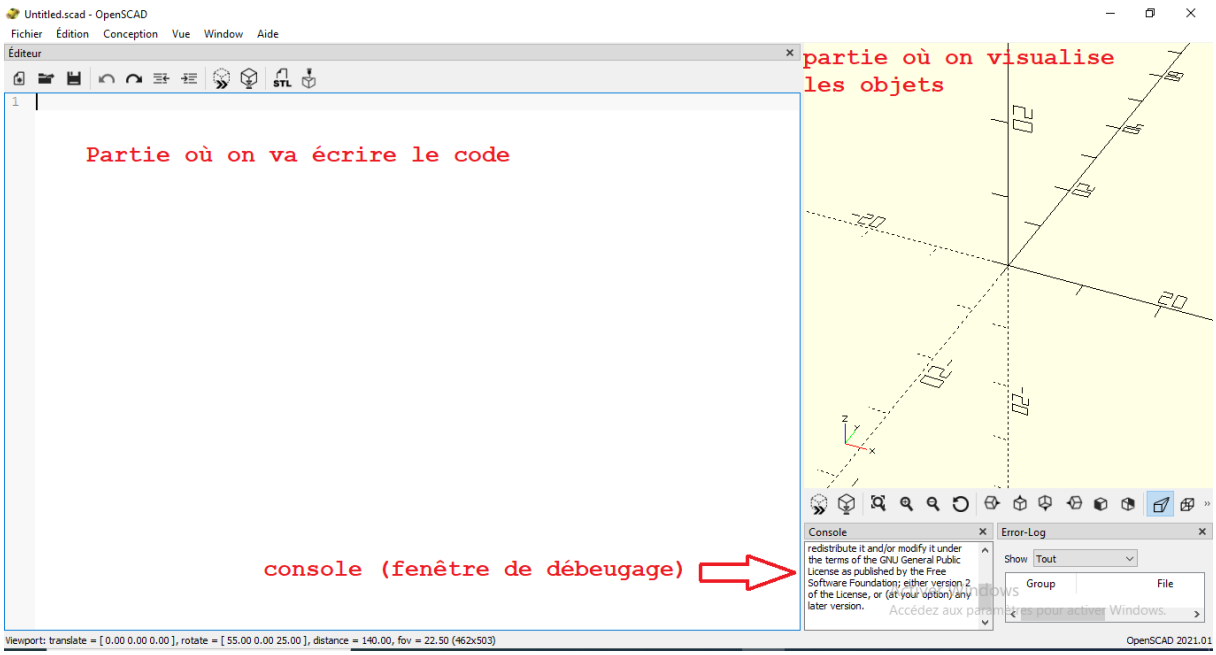
L'avantage principal du logiciel est la possibilité de rendre ces objets parfaitement paramétrables en fonction des variables crées.

OpenSCAD est un logiciel gratuit. Pour le télécharger, allez sur : http://openscad.org/ → downloads et choisissez la version compatible avec votre système d'exploitation.

Maintenant que vous avez installé openSCAD sur votre PC, commençons par apprendre les bases du logiciel.

Tutoriel 1 : les Bases et formes primitives

Lorsque vous ouvrez le logiciel pour la 1ere fois, c'est ce que vous devez voir comme fenêtre :



Créons maintenant notre 1ere forme.

Sphère

Nous allons créer une sphère. Pour cela nous allons écrire le code suivant

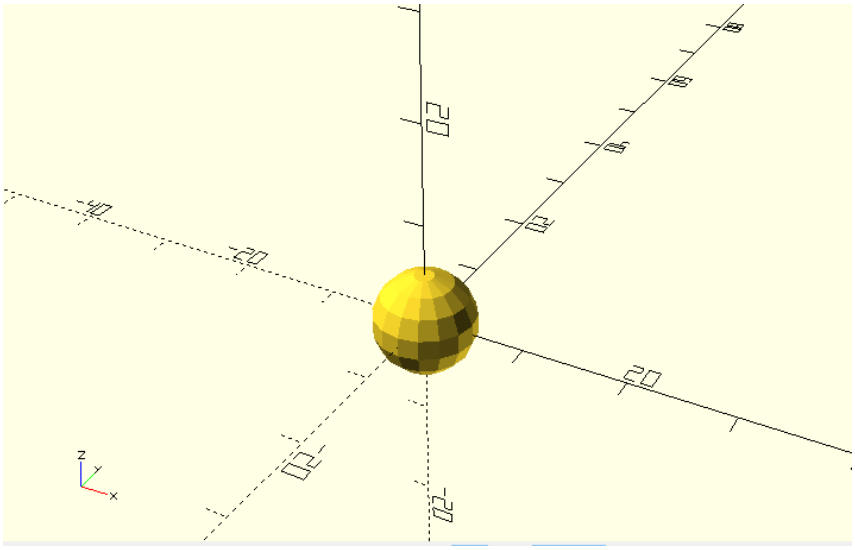
```
1  sphere(r=5);
```

La fonction sphère prend comme argument le rayon r de la sphère. Nous avons choisi r=5mm.

NB : n'oubliez pas de mettre le « ; » à la fin de l'instruction sinon le code ne fonctionnera pas malheureusement.

Pour afficher l'objet (sphère) nous devons compiler le programme en appuyant sur la touche F5 sur le clavier ou bien en appuyant sur l'icône « calculer l'aperçu » sur la barre d'outils.

Après compilation du code, la sphère s'affiche.



- Pour zoomer vous pouvez utiliser la roulette de la souris.
- Pour pouvoir faire une rotation de l'objet visualisé, vous devez cliquer et maintenir le bouton gauche de la souris et faire tourner l'objet dans le sens souhaité.
- Pour déplacer l'objet dans l'espace, faites clique droit et déplacer l'objet vers la position souhaité (pour pouvoir visualiser d'autres objet si on en a plusieurs par exemple)

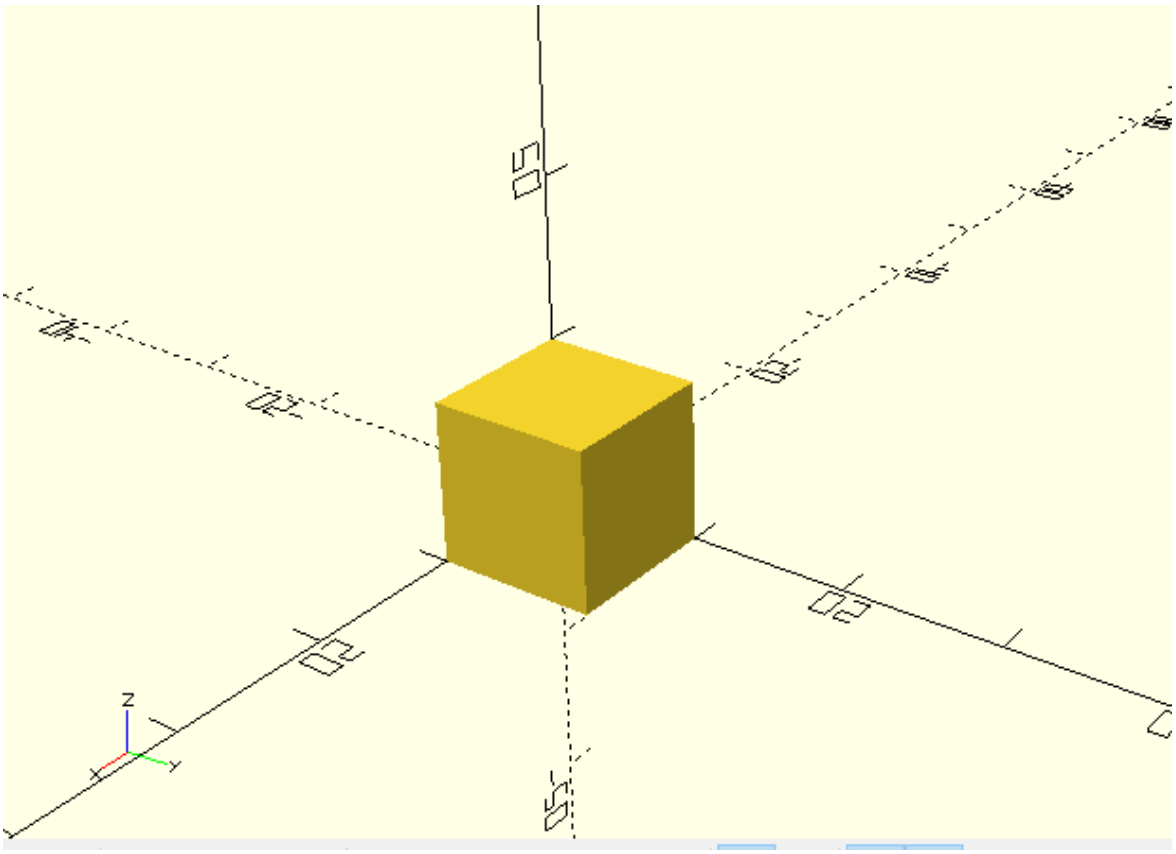
Cube

Pour créer un cube, on utilise la fonction cube() qui prend comme argument la taille du cube (son arête) ou les valeurs largeur, profondeur, et hauteur).

```
1  cube(size=10);
```

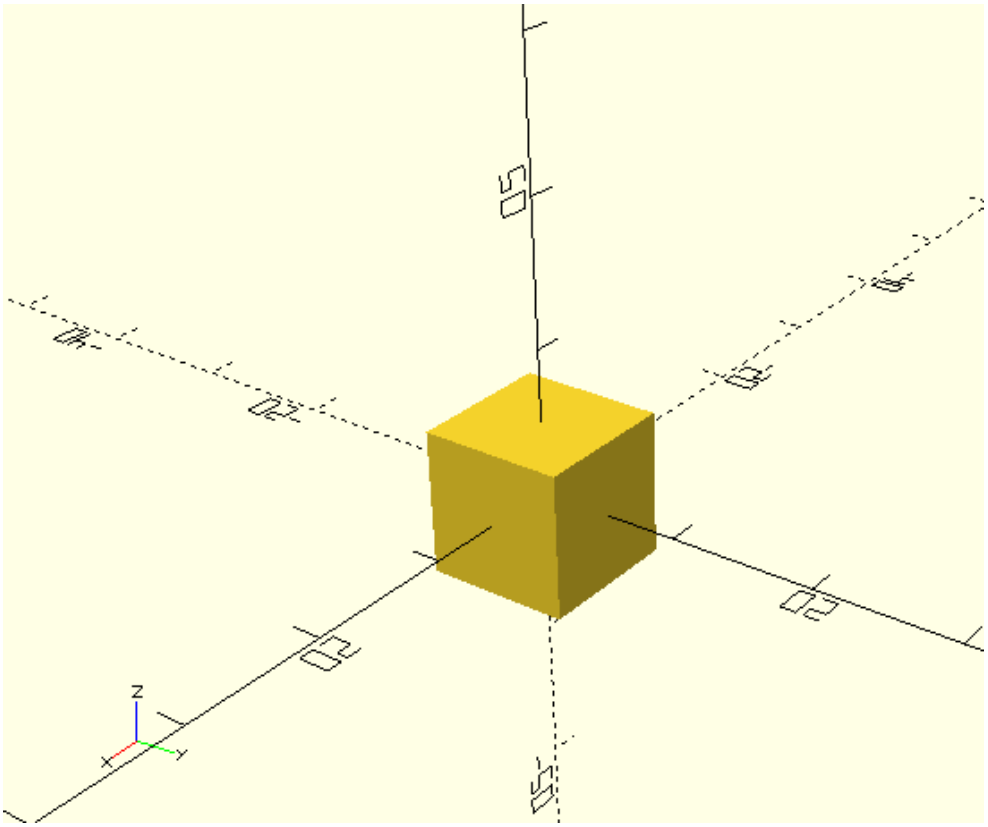
```
1  cube([10,10,10]);
```

Les 2 codes affichent le même objet, c'est cube d'arête =10



Maintenant, si on veut centrer le cube on ajoute l'argument « center=true » à la fonction cube().

```
1  cube([10,10,10],center=true);
```



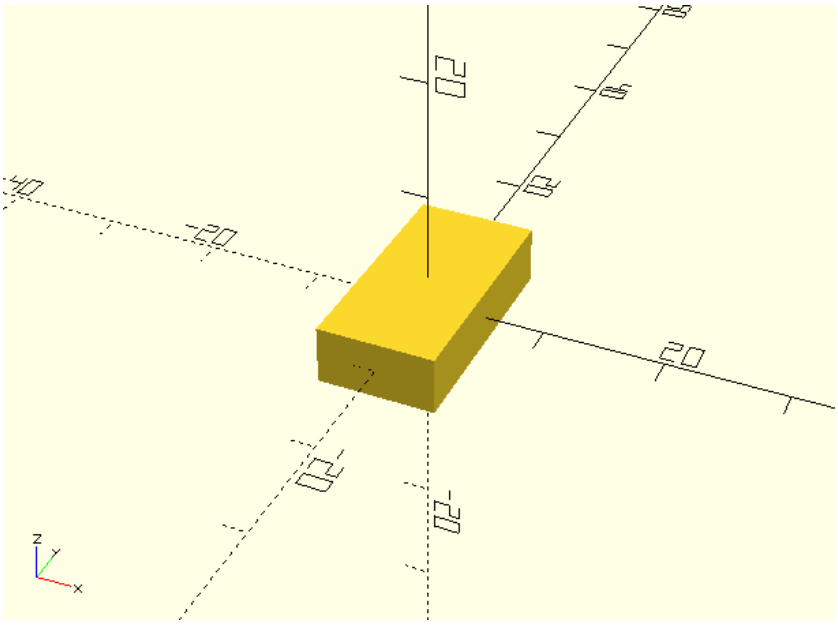
Le cube maintenant est positionné sur le centre

Rectangle

Maintenant qu'on sait comment créer un cube on peut facilement afficher un parallélépipède.

On va juste modifier les valeurs de la largeur, la profondeur, et la hauteur dans la fonction cube().

```
1  cube([10,20,5],center=true);
```

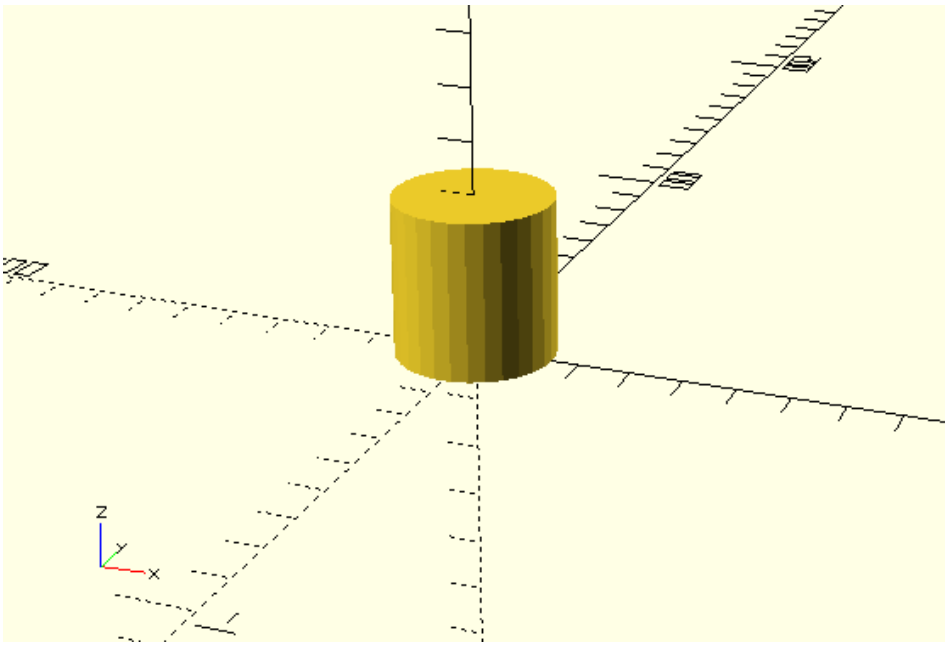


Cylindre

Pour créer un cylindre, on tape la fonction cylinder().

Elle prend la hauteur du cylindre comme 1ere valeur et son rayon r ou diamètre d comme 2e valeur.

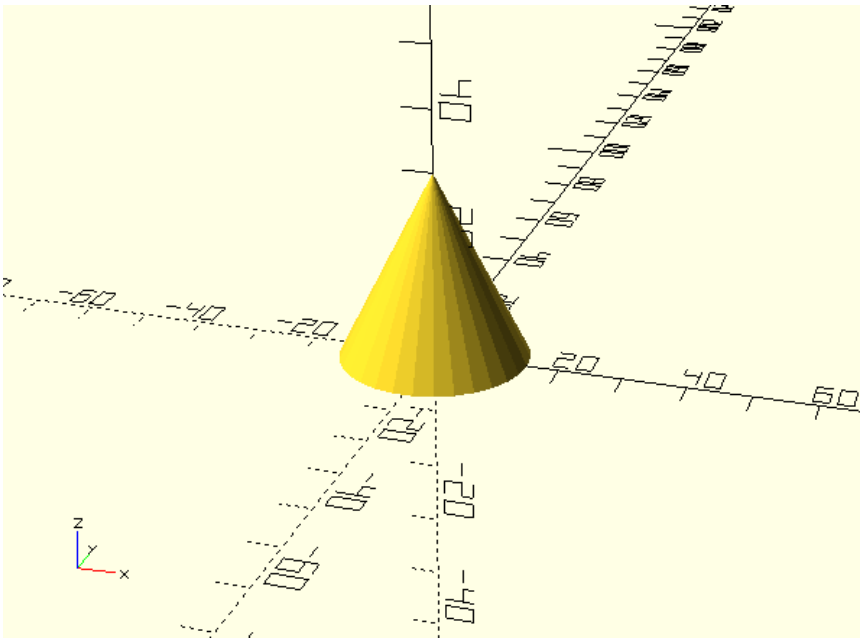
```
1  cylinder(30,r=15);
```



Cône

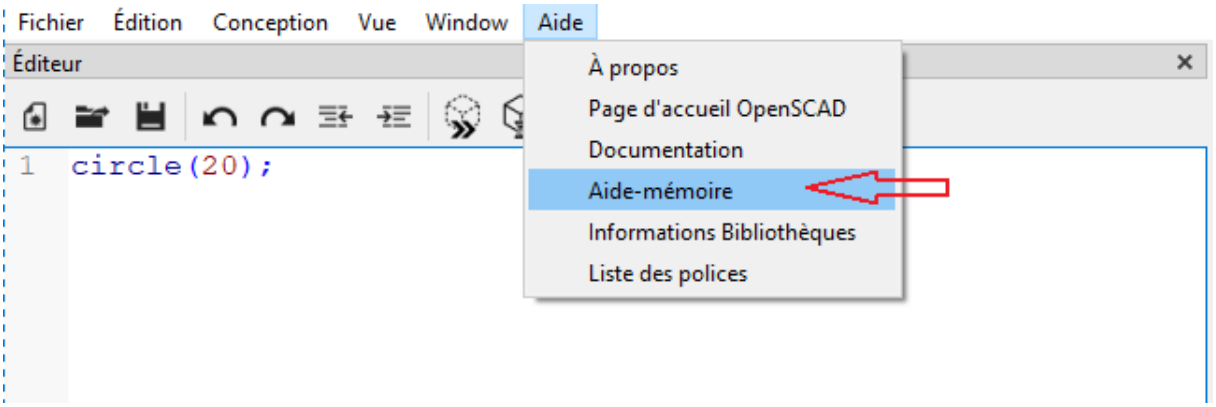
On peut maintenant à partir de la fonction cylinder de créer un cône en mettant 2 valeurs différentes du rayon.

```
1  cylinder(30,r1=15,r2=0);
```



On arrive à la fin de ce premier tutoriel.

NB : si vous voulez savoir plus de détail sur une fonction quelconque cliquer sur aide → aide-mémoire comme c'est montré dans la figure suivante :



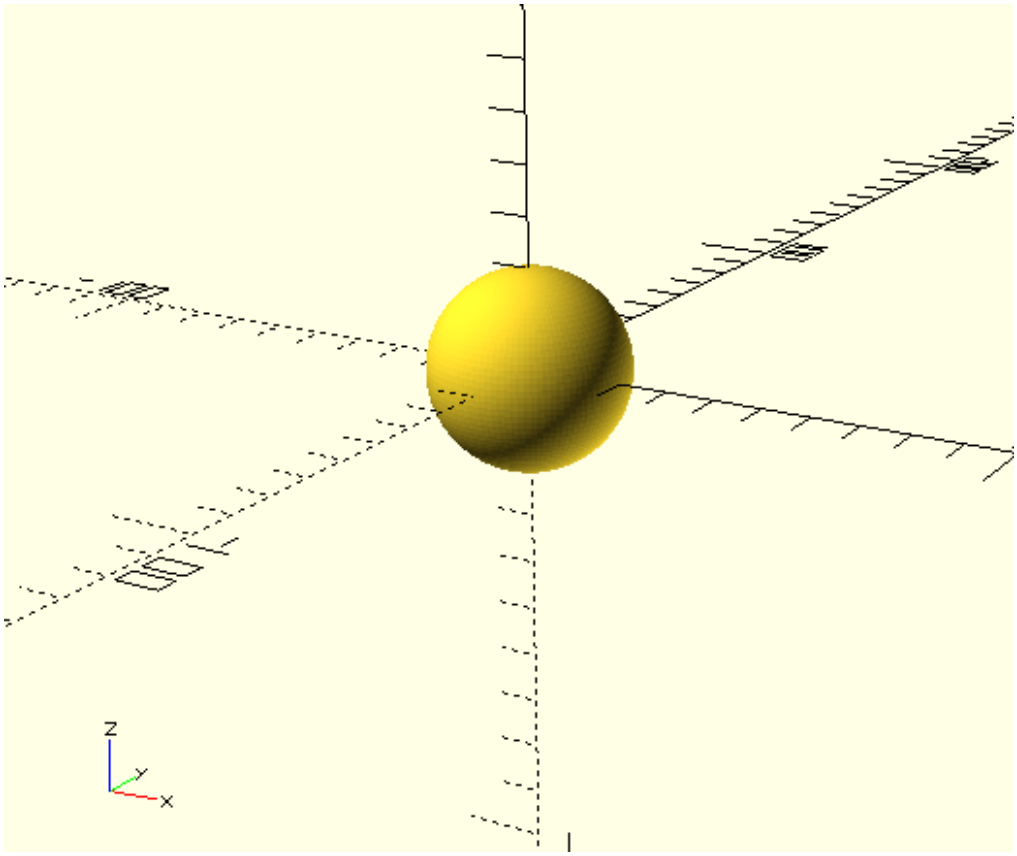
Tutoriel 2 : résolution, console, variable

Résolution

Comme vous avez remarqué dans le tutoriel 1, les formes affichées avait une résolution pas trop importante car le nombre de face qui la constitue était faible.

Pour remédier à ce problème on va utiliser « $fn » qui détermine la résolution de la forme visualisée, c'est-à-dire le nombre de fragments constituant l'objet.

```
1  sphere(r=20,$fn=100);
```

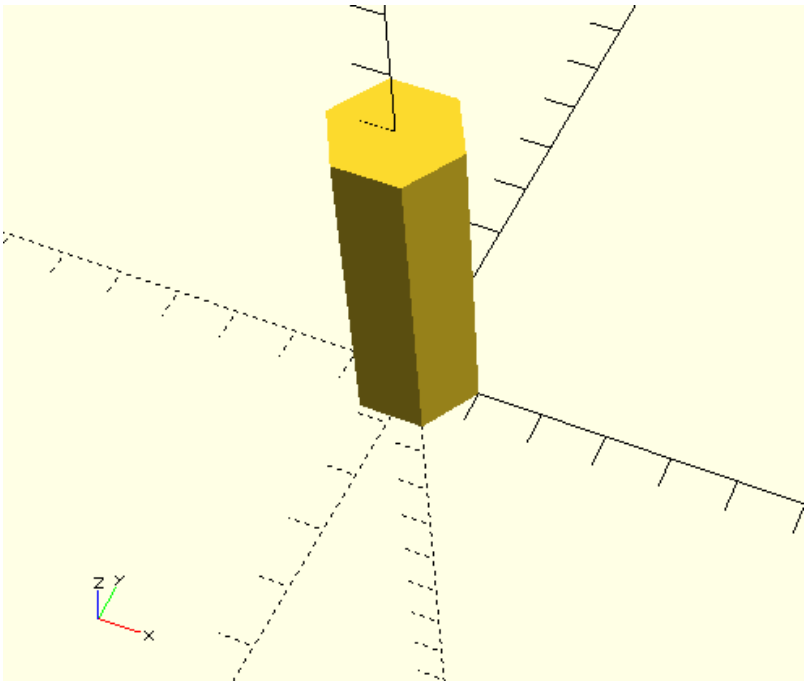


La sphère maintenant a clairement une meilleure résolution

Attention, si vous augmenter trop la résolution, le temps de calcul sera plus grand. Essayez de choisir des valeurs comprises entre 50 et 200.

La résolution peut être utile aussi pour afficher des polygones. Par exemple, si on veut afficher un hexagone on règle la valeur de la résolution à 6.

```
1  cylinder(50,r=10,$fn=6);
```



Variables et console

En informatique, les variables sont des symboles qui associent un nom à une valeur.

Pour créer une variable, on choisit un nom à cette variable et on met une valeur dedans.

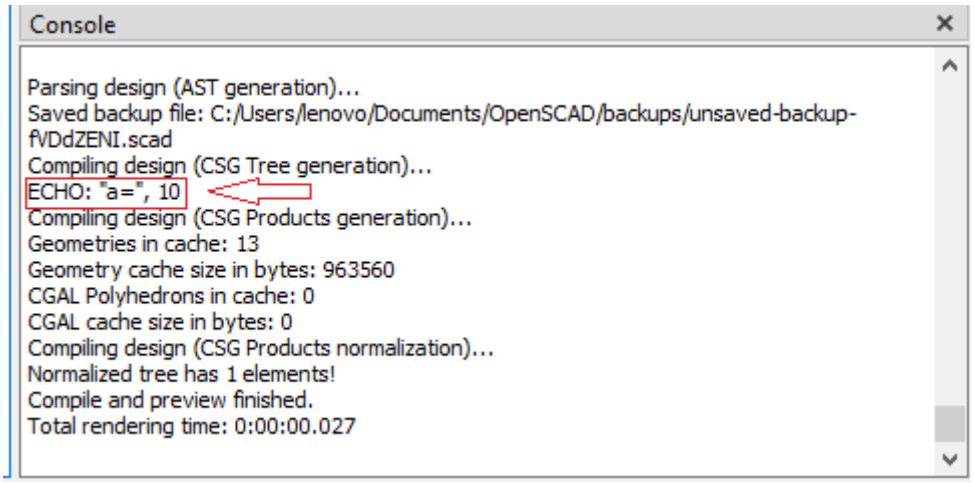
```
1  a=10;  //variable a
```

La variable 'a' contient la valeur 10.

NB : l'expression « variable a » en bleu représente un commentaire. On peut l'écrire tapant « // ».

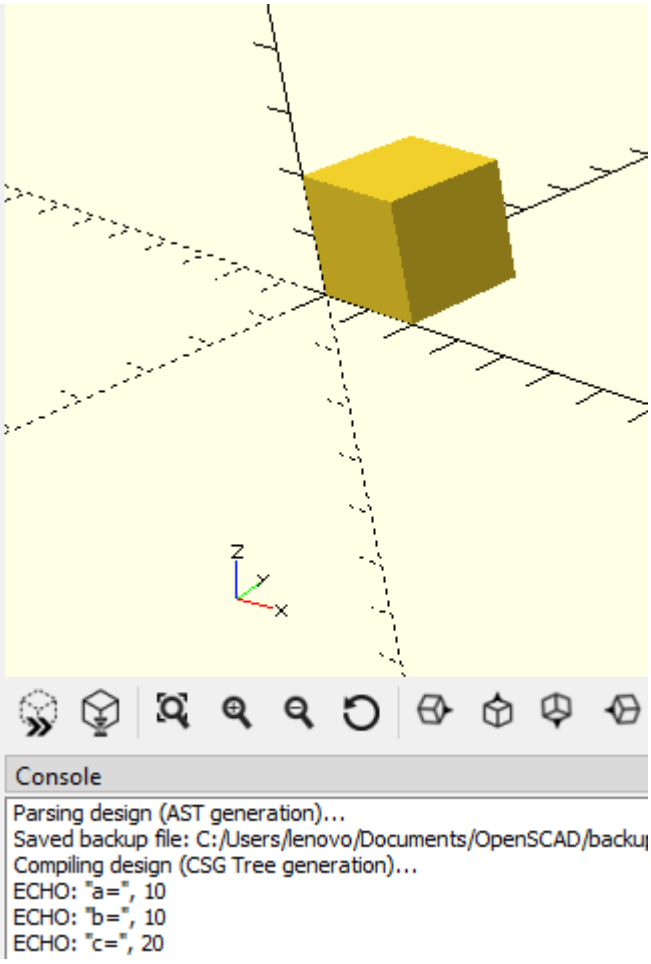
Pour afficher la valeur de la variable a dans la console, on utilise la fonction écho

```
1  a=10;
2  echo("a=",a);
```



Je vous laisse le soin de lire et comprendre le code suivant qui utilise 3 variables a, b et c.

```
1  a=10;
2  echo("a=",a);
3  b=10;
4  echo("b=",b);
5  c=a+b;
6  echo("c=",c);
7  cube(c);
```



Tutoriel 3 : translate, rotate

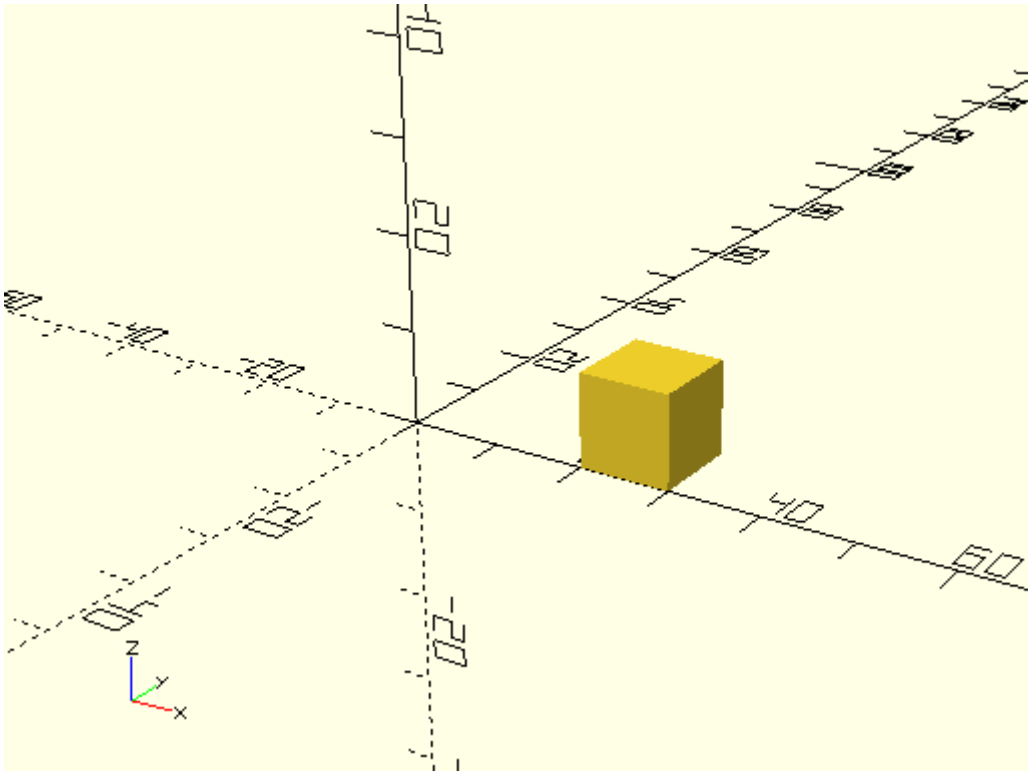
Translate

Pour translater un objet on utilise la fonction translate([x,y,z]) avant l'objet à translater.

La translation peut se faire sur x et/ou y et/ou z et les x,y et z peuvent prendre des valeurs positives et négatives aussi.

```
1  translate([20,0,0]) cube(10);
```

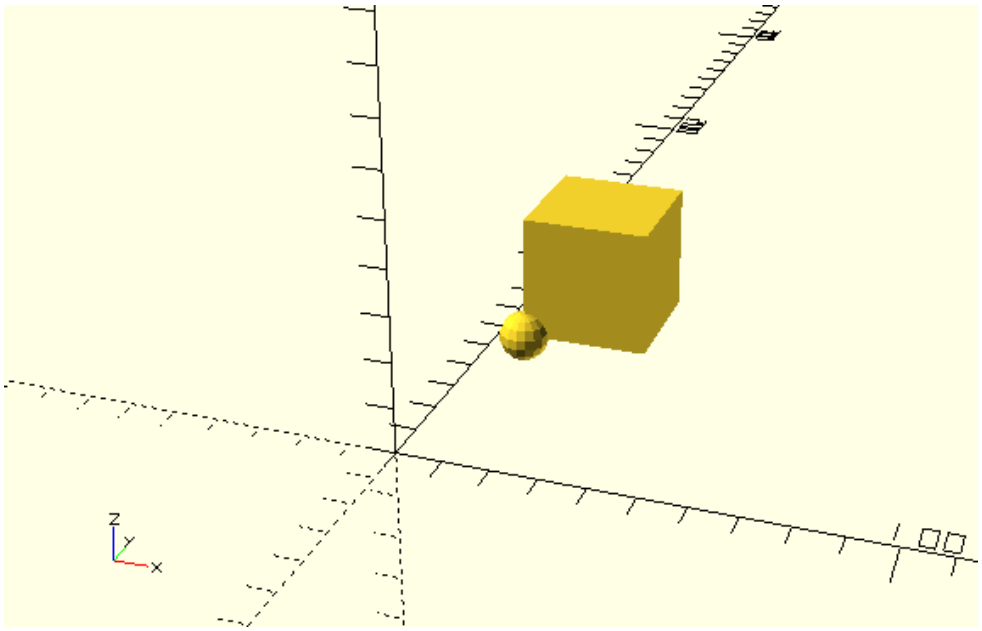
Cette instruction permet donc de faire une translation sur l'axe des x d'un cube d'arête 10



On peut aussi translater un groupe d'objet à la fois en mettant les objets à l'intérieur de deux accolades.

```
1 translate([20,20,20]){
2     cube(25);
3     sphere(r=5,center=true);
4
5 }
```

Cette instruction permet donc de translater d'une valeur de 20 sur l'axe x, y et z à la fois d'un cube d'arête 25 et d'une sphère de rayon r centrée.

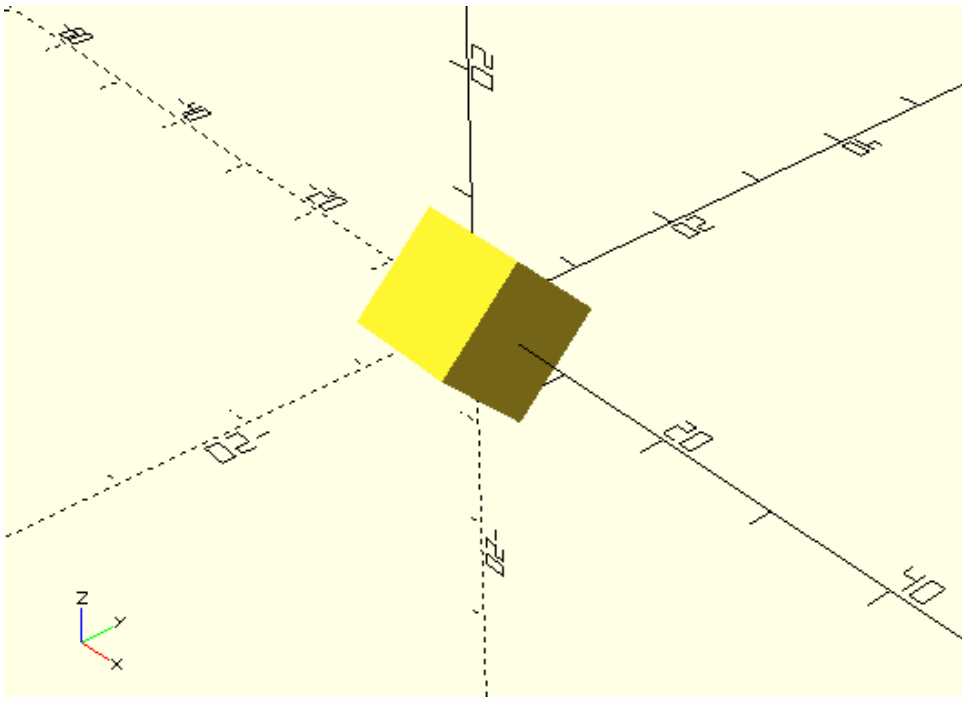


Rotate

De la même logique que la translation, on peut utiliser rotate([x,y,z]) avec x , y et z qui représentent les valeurs en degré de l'angle de rotation.

Pour faire une rotation de 45 degrés sur l'axe x d'un cube d'arête 10 centré :

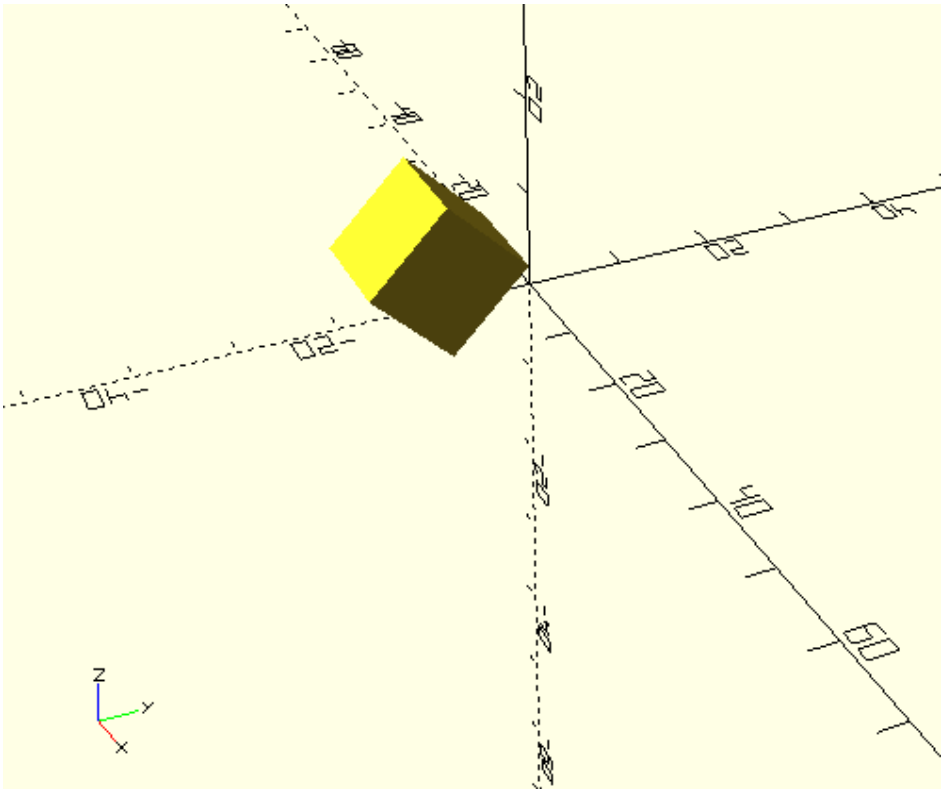
```
1  rotate([45,0,0]) cube(10,center=true);
```



Maintenant que vous savez utiliser translate() et rotate(), vous pouvez les utiliser à la fois sur un objet.

```
1  rotate([45,0,0]) translate([20,0,25]) cube(10,center=true);
```

Cette instruction permet de faire une rotation sur x du cube puis une translation du même cube sur x et z avec les valeurs 20 et 25 respectivement.



Amuser vous à faire des translations et des rotations avant de passer au tutoriel suivant.

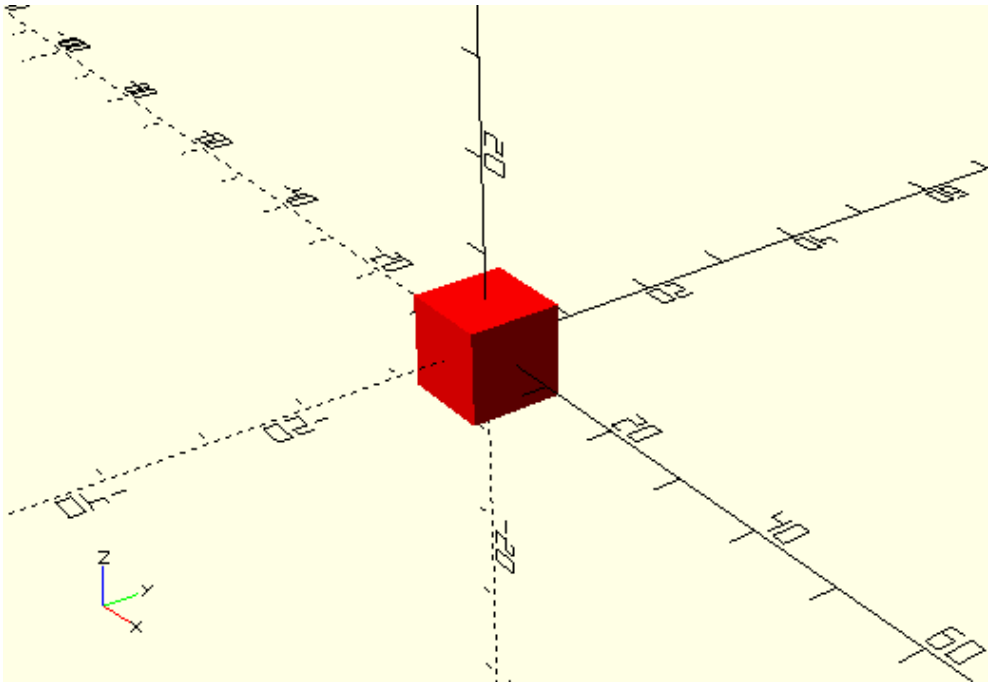
Tutoriel 4 : color, mirror

Color

Pour colorier un objet on utilise color([R,G,B]),avec R, G et B les codes décimal des couleur.

Par exemple pour afficher un cube d'arête 10 en rouge, on tape :

```
1  color ([1,0,0])cube(10,center=true);
```

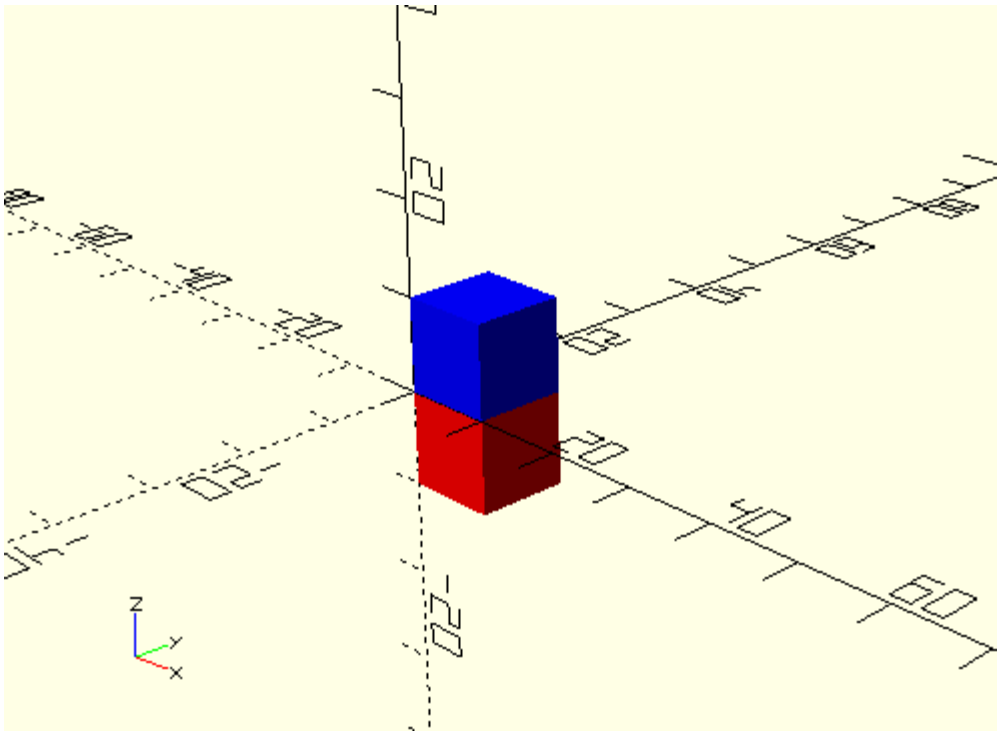


Mirror

À chaque axe x, y ou z on va devoir renseigner soit 0 soit 1. 1 pour indiquer que le miroir est appliqué et 0 pour veut dire l'inverse.

Pour appliquer un miroir sur l'axe z par exemple sur un cube :

```
1  color([0,0,1]) cube(10);
2  mirror ([0,0,1])color([1,0,0])cube(10);
3
```

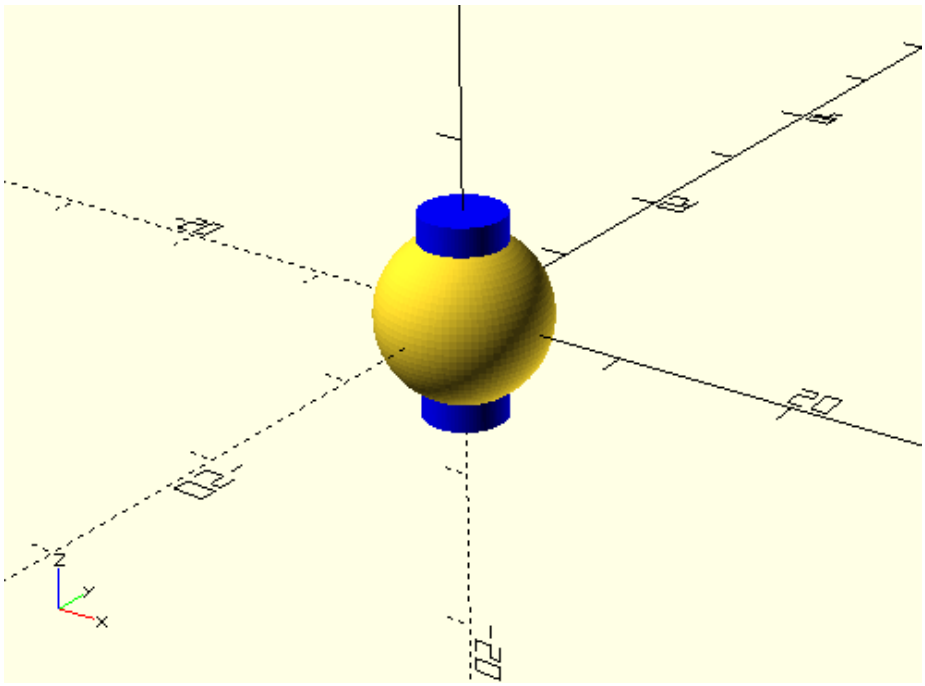

Z
Y
X

Tutoriel 5 : difference, intersection et union

Difference

On veut faire la différence entre une sphère et un cylindre, c'est-à-dire qu'on va créer un trou débouchant dans la sphère avec un rayon égal au rayon du cylindre.Pour cela on utilise la fonction difference().

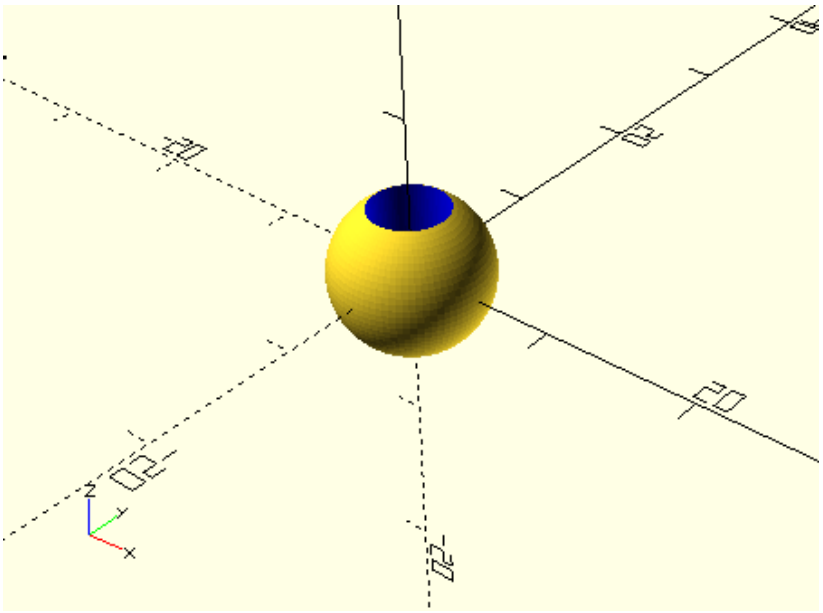
Créons d'abord la sphère et le cylindre avec une même résolution R.



Maintenant, si on met ce code à l'intérieur de la fonction difference(), on va pouvoir éliminer la partie cylindre de la sphère.

```
1  R=80;
2 difference(){
3  sphere(r=5,$fn=R);
4  color([0,0,1]) cylinder(h=12,r=2.5,$fn=R,center=true);
5 }
```

En effet la 1ere instruction c'est la valeur qui va rester, la 2e instruction ou les suivantes seront soustraites à la 1ere.



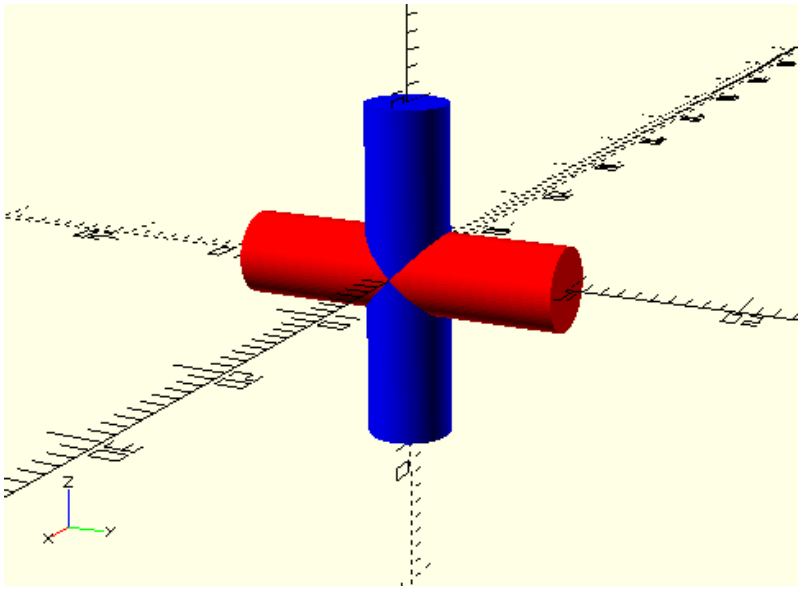
Comme vous voyez, notre cylindrer a bien été soustrait de la sphère.

Intersection

La fonction intersection() permet de conserver uniquement les parties communes à une ou plusieurs formes.

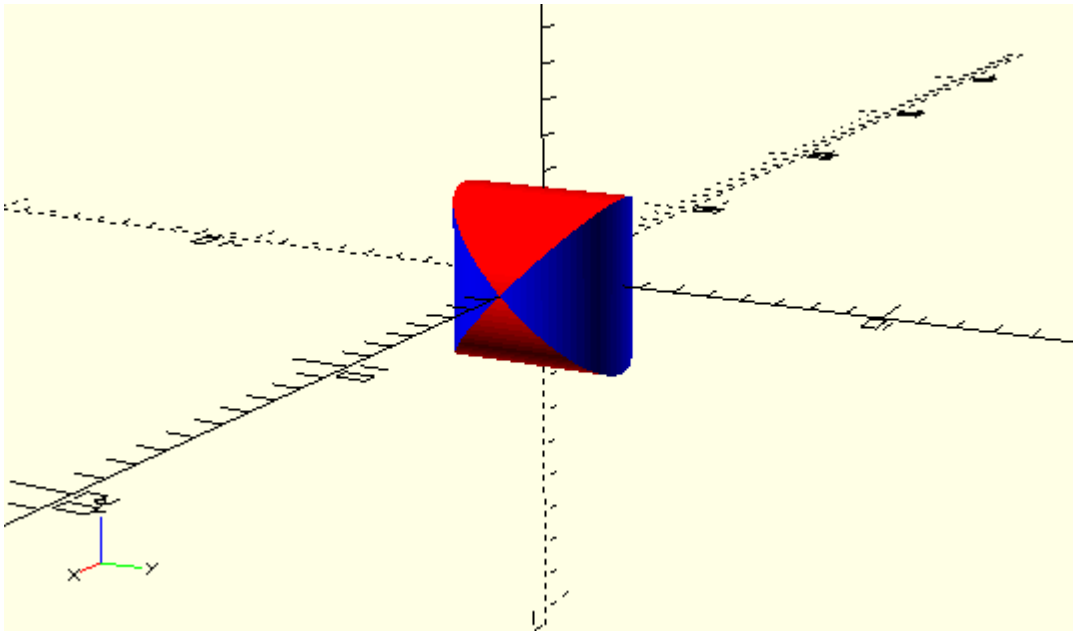
Créons 2 cylindres de mêmes dimensions, l'un posé horizontalement (rotate 90 degres sur l'axe des x) et l'autre verticalement.

```
1  R=80;
2  color([0,0,1]) cylinder(h=20,r=2.5,$fn=R,center=true);
3  color([1,0,0]) rotate([90,0,0])cylinder(h=20,r=2.5,$fn=R,center=true);
```



Maintenant mettons ce code à l'intérieur de la fonction intersection().

```
R=80;
intersection(){
color([0,0,1]) cylinder(h=20,r=2.5,$fn=R,center=true);
color([1,0,0]) rotate([90,0,0])cylinder(h=20,r=2.5,$fn=R,center=true);
}
```



Comme vous voyez sur la figure, il ne reste que la partie commune entre les 2 cylindres.

Union

Comme son nom l'indique, la fonction union() permet d'unir des formes à fin de former une nouvelle. En modélisation 3D, deux formes qui se touchent ne forme pas une seule.

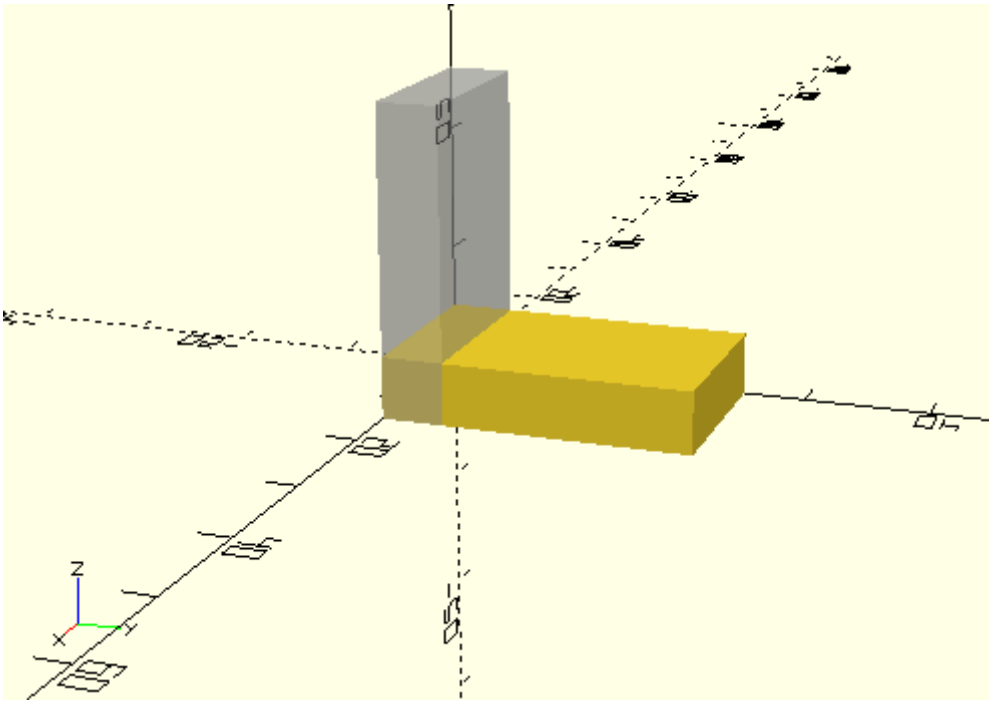
Créons par exemple 2 parallélépipèdes rectangles avec 1 partie commune.

```
//*desactiver un objet
//!voir objet uniquement
//#surligner
//%transparence

R=30;
cube([15,25,5]);
%translate([0,5,0])rotate([90,0,0])cube([15,25,5]);
```

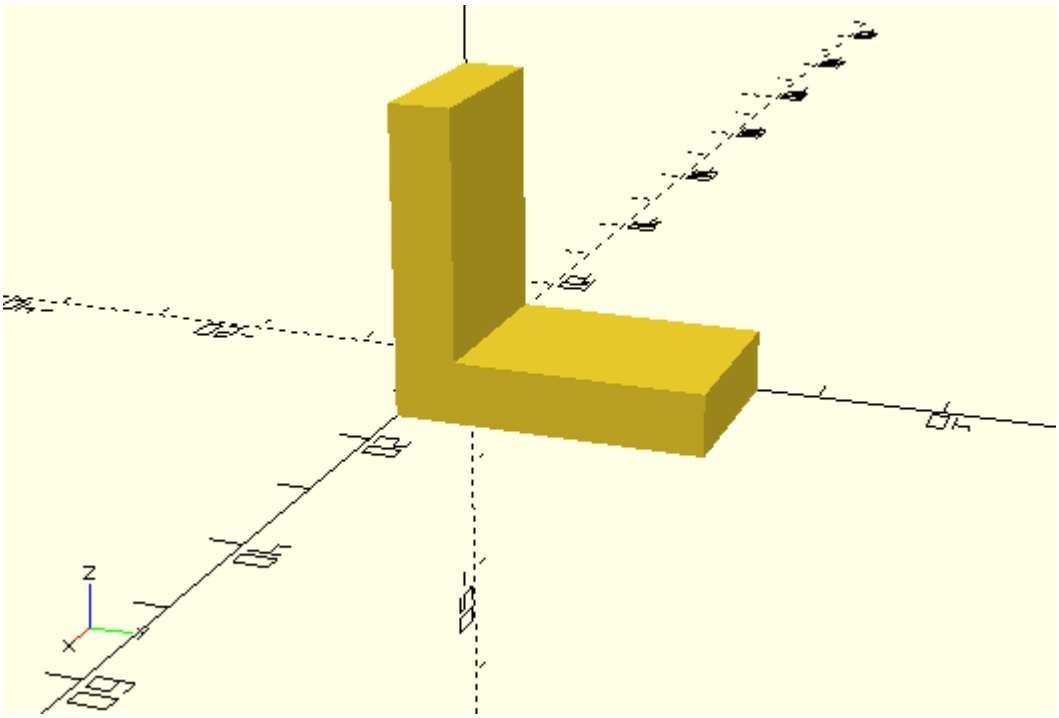
Ici on a utilisé le symbole « % » avant le 2e cube. C'est une opération booléienne qui permet de rendre transparence la forme.

NB : d'autre opérations sont écrites en commentaires en haut du code. Vous pouvez les utiliser sur vos formes.



Maintenant on va utiliser la fonction union() pour unir les 2 formes et obtenir une seule.

```
5 union(){
6  R=30;
7  cube([15,25,5]);
8  translate([0,5,0])rotate([90,0,0])cube([15,25,5]);
9 }
```



Maintenant si vous exportez ceci, vous n’aurez qu’un seul et unique objet.