Formation Impression 3d (pour les mathématiques)

OpenSCAD

Gérald Bonin

18 novembre 2026

Objectifs de la formation :

- Utiliser des logiciels pour créer des objets 3D,
- Configurer et utiliser une imprimante 3D,
- Donner des pistes d'activités élèves utilisant des objets 3D,
- Partager des activités utilisant des objets 3D.

Pourquoi concevoir et imprimer des objets 3d ?

- Pour diversifier les cours et ainsi relancer l'attention et l'engagement des élèves,
- Pour aider les élèves qui ont plus de mal avec l'abstraction, sur papier ou en vrai ce n'est pas pareil,
- Comme supports d'activités supplémentaires et pour donner du sens . . .

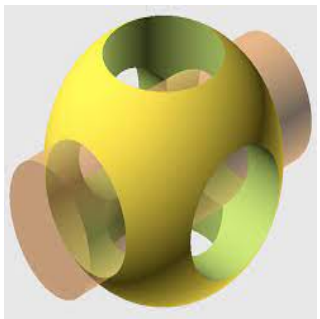
Objets utilisés en classe :

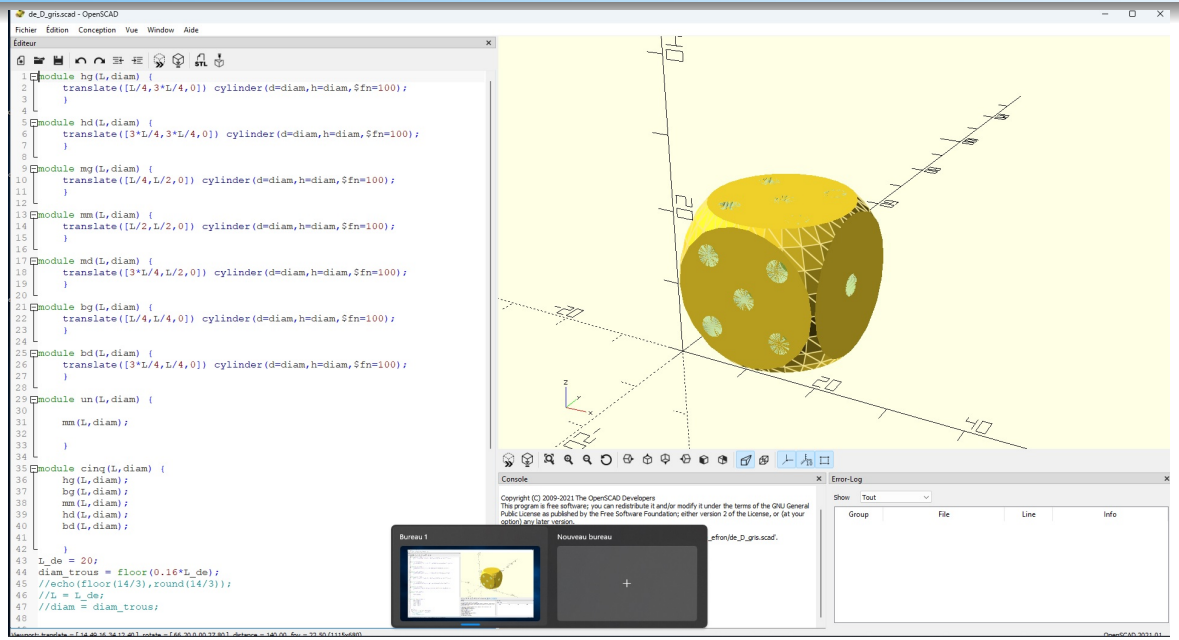


Quelle imprimante 3d choisir ?

- Pour tous les budgets de 200 € environ à plusieurs milliers d'euros,
- Choisir des modèles éprouvés, avec une communauté large
- On trouve de nombreux tests sur internet : lien 1, lien 2
- Ne pas confondre avec les imprimantes à résine !

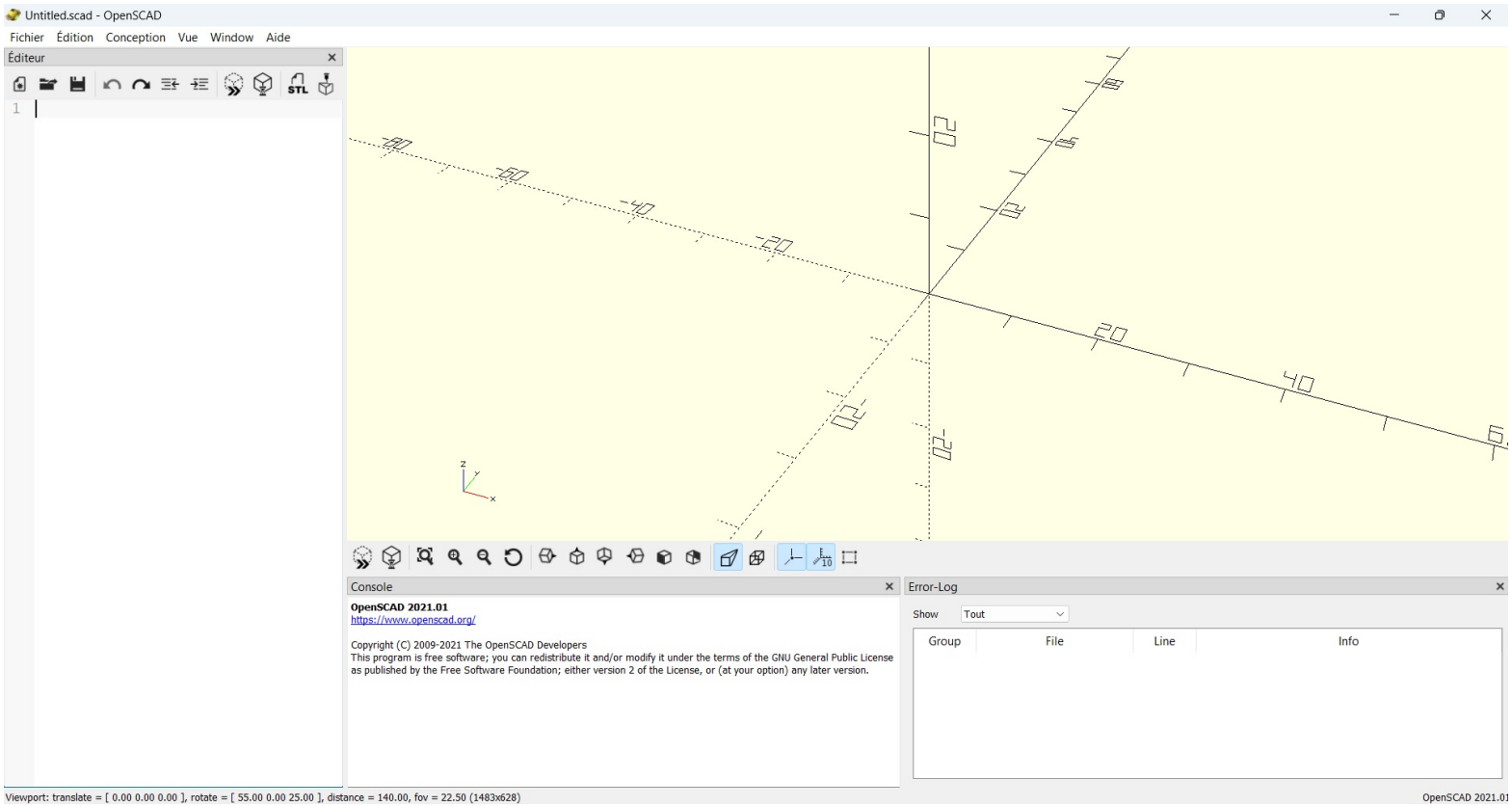
OpenSCAD





- open source
- performant et relativement complet
- objets créés à l'aide de scripts (bien adapté aux professeurs de mathématiques)
- lien pour téléchargement :
  https://openscad.org/downloads.html

Présentation de l'interface

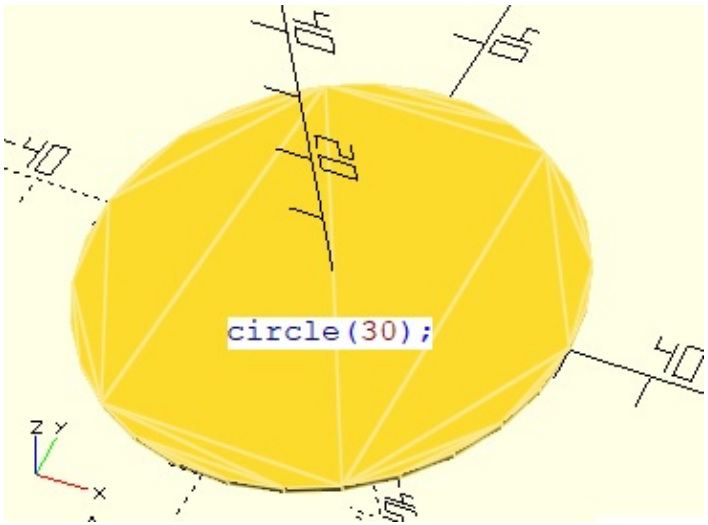


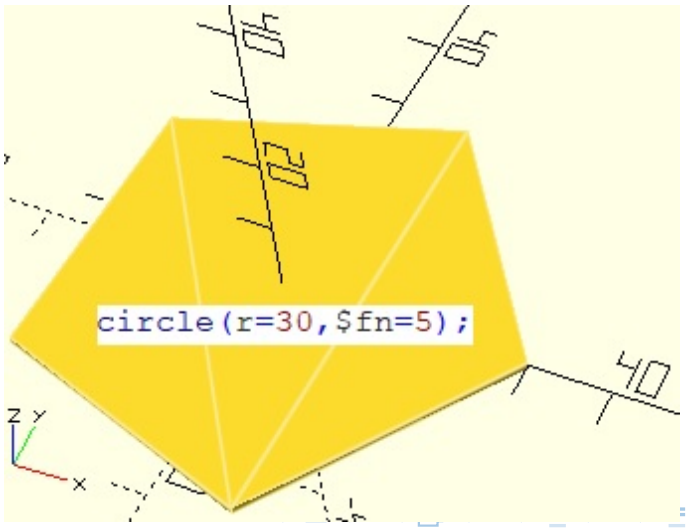
- à gauche : fenêtre script
- à droite : repère 3d, clic gauche pour rotations, clic droit pour déplacer, unités en mm

les formes 2d prédéfinies : circle(rayon)

circle(rayon,$fn)

- Disque centré sur l'origine et contenu dans le plan xOy
- $fn : optionnel, nombre de subdivisions, Plus précisément, on obtient un polygone régulier de $fn côtés.
- les instructions circle(30) ; ou circle(r=30) ; ou circle(d=60) ; produisent le même résultat.

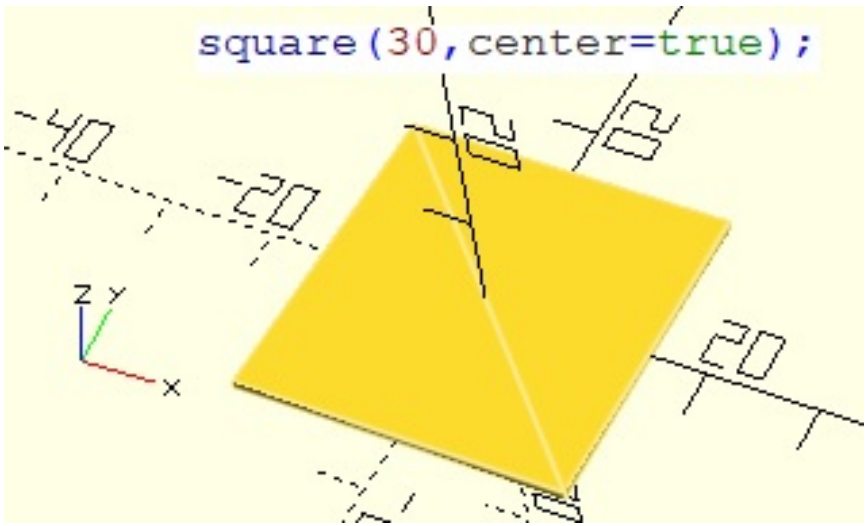


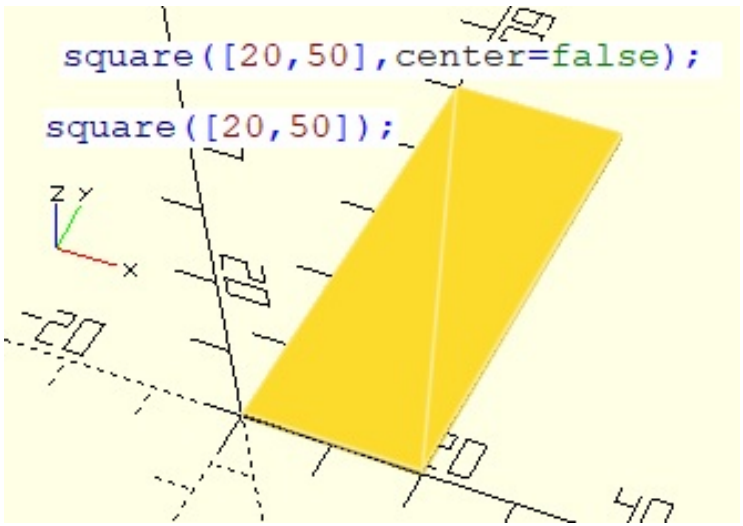


les formes 2d prédéfinies : square([x,y])

square([x,y],center)

- Pour obtenir un rectangle centré sur l'origine (center = true) ou dont un sommet est l'origine (center = false, par défaut)
- x et y sont les dimensions du rectangle
- square(x) ; ou square([x,x]) ; produisent un carré de côté x.





Construction de la base du solide point par point

polygon(points = [[x, y], ...], paths = [[p1, p2, p3..], ...]) ;

permet de placer les points de coordonnées (x1,y1), (x2,y2)... afin de former la base du solide, les points seront reliés dans l'ordre défini par path.

Par exemple, les scripts

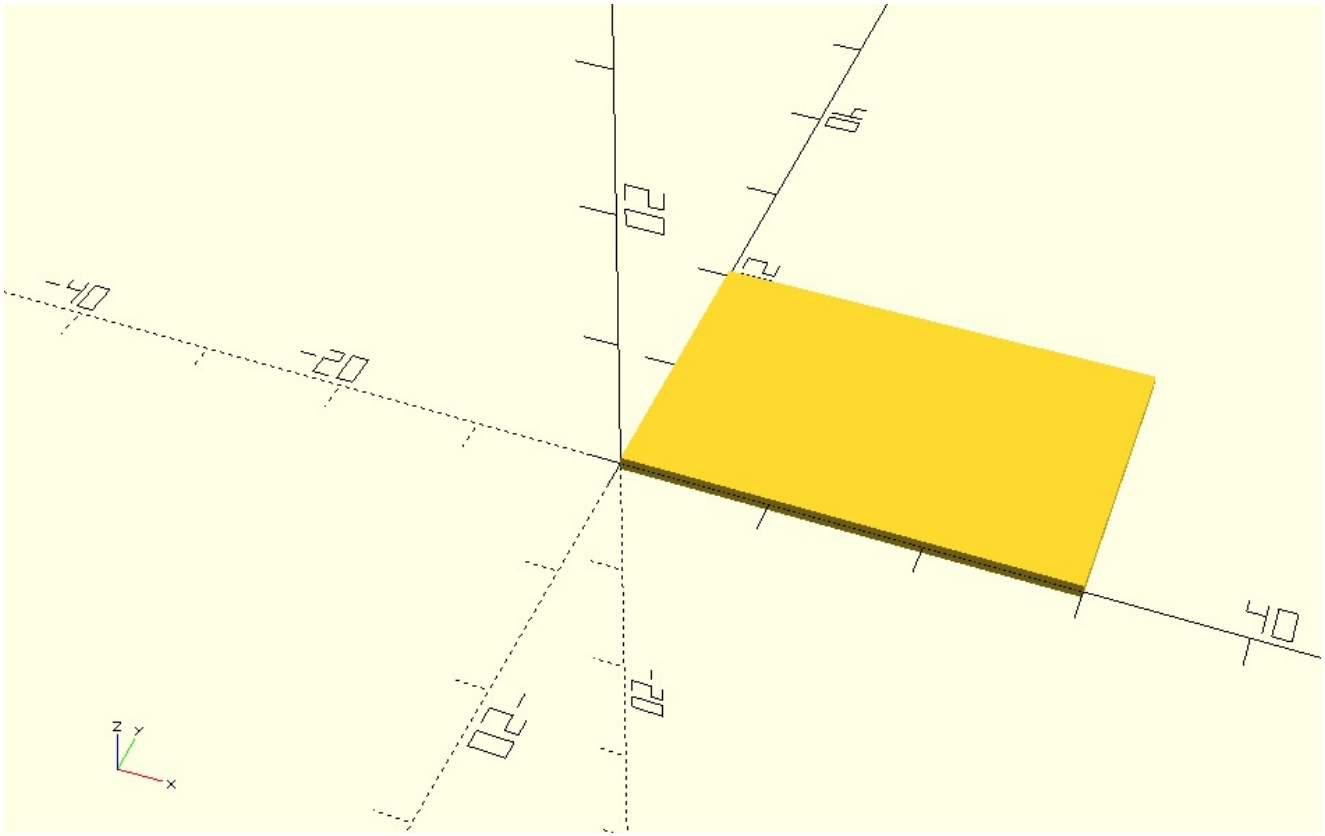
```
1  polygon([[0,0],[30,0],[30,20],[0,20]],[[0,1,2,3]]);
```

ou

```
1  polygon([[0,0],[30,0],[30,20],[0,20]]);
```

Construction de la base du solide point par point

produisent après appui sur la touche  :

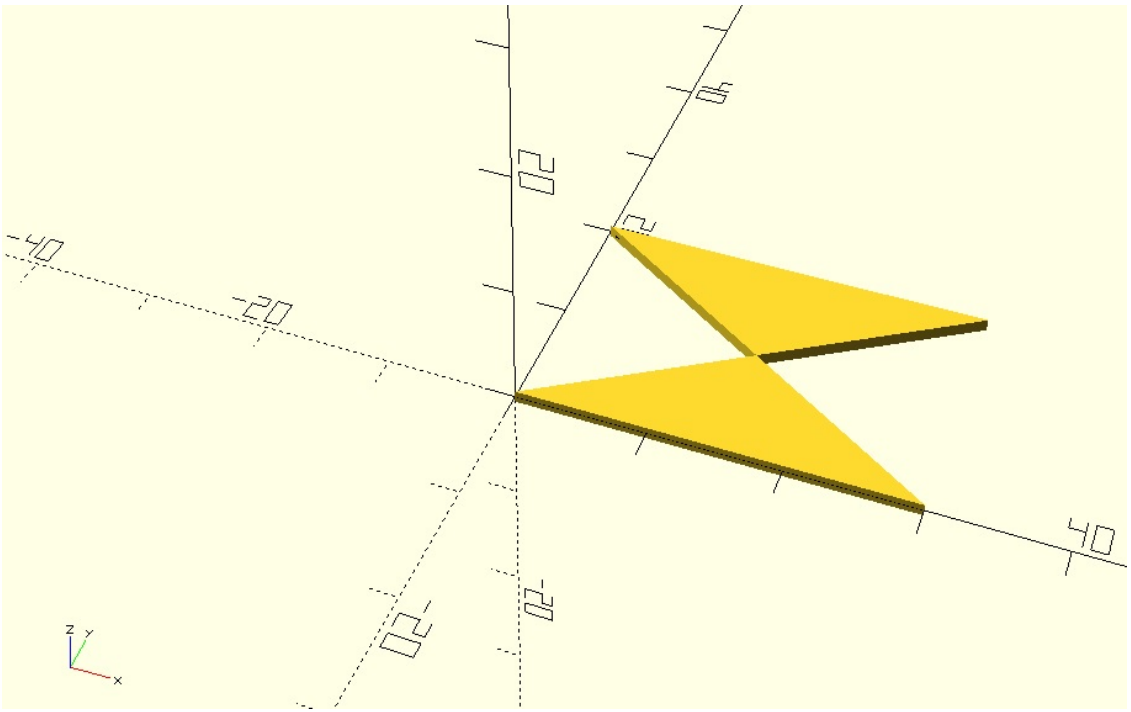


Construction de la base du solide point par point

tandis que

```
1  polygon([[0,0],[30,0],[30,20],[0,20]],[[0,2,3,1]]);
```

produit :



Extrusion linéaire

Sur Openscad les scripts sont structurés ainsi :

```
1  action1()    action2() objet();
```

ou

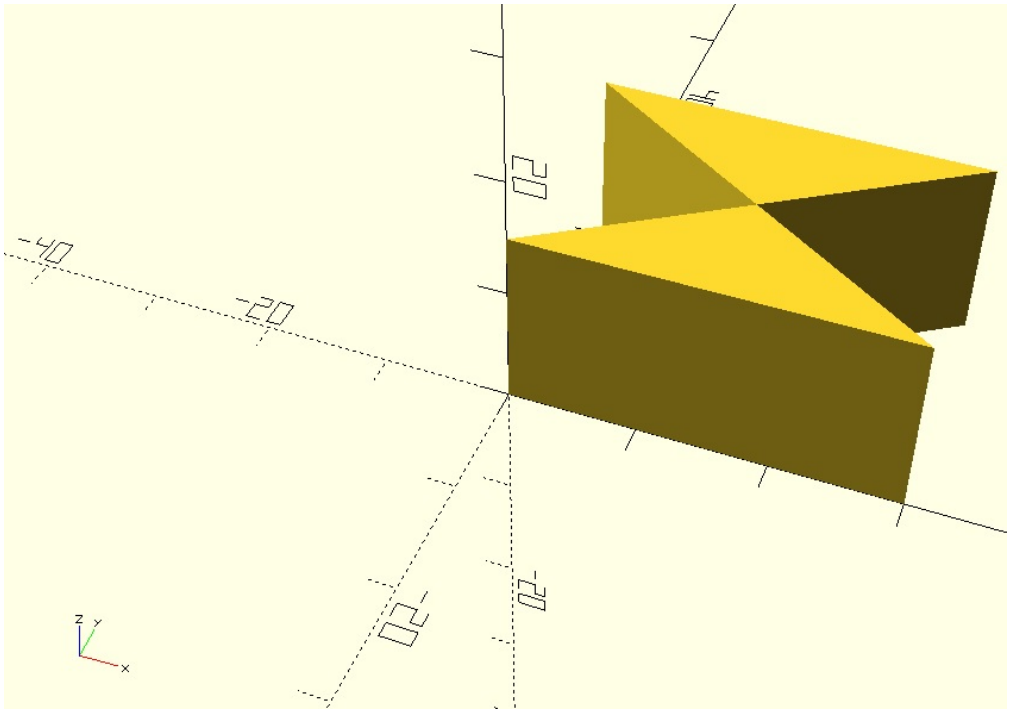
```
1  action() {
2    action1() objet1();
3    action2() objet2();
4  }
```

Ainsi le script, qu'on peut écrire sur une seule ligne :

```
1  linear_extrude(15)
2  polygon([[0,0],[30,0],[30,20],[0,20]],[[0,2,3,1]]);
```

produit :

```
1  linear_extrude(15)
2  polygon([[0,0],[30,0],[30,20],[0,20]],[[0,2,3,1]]);
```



Activité 1

Construire le trapèze du puzzle de Lewis Caroll, avec une hauteur de 5mm, puis produire le fichier près à être imprimé au format gcode.
Lien pour imprimer l'objet obtenu

Extrusion linéaire

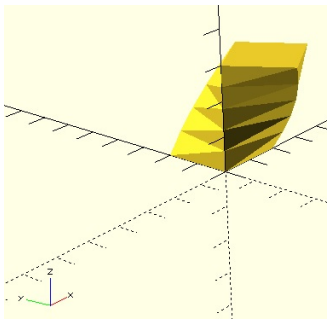


linear_extrude(height,center,convexity,twist,slices,scale,$fn) {...}

- height : hauteur de l'extrusion
- center : *true* ou *false*, optionnel, centrer verticalement l'extrusion
- twist : option, angle rotation d'axe (Oz) entre base et sommet
- scale : optionnel, rapport dimensions entre base et sommet
- $fn : optionnel,nombre de subdivisions pour les faces latérales (lissage)

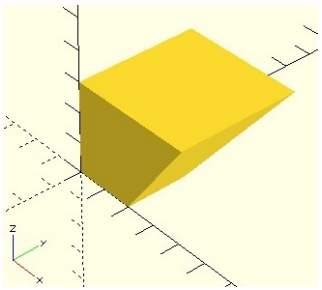
Extrusion linéaire : exemple twist

```
1 linear_extrude(30,center=false,twist=60)
2 polygon([[0,0],[20,0],[20,20],[0,20]]);
```

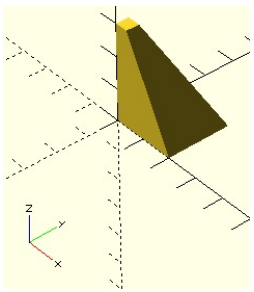


Extrusion linéaire : exemples scale

```
1 linear_extrude(30, scale=2)
2 polygon([[0,0],[20,0],[20,20],[0,20]]);
```



```
1 linear_extrude(30, scale=1/5)
2 polygon([[0,0],[20,0],[20,20],[0,20]]);
```



Construction d'un solide point par point

polyhedron(points = [[x1,y1,z1], [x2,y2,z2],...], faces = [[$point_1$, $point_2$,...],[$point_1$, $point_4$,...], ...]) ;

permet de construire un polyhèdre à partir d'une liste de points et d'une liste de faces (décrites par des listes d'indices de points)

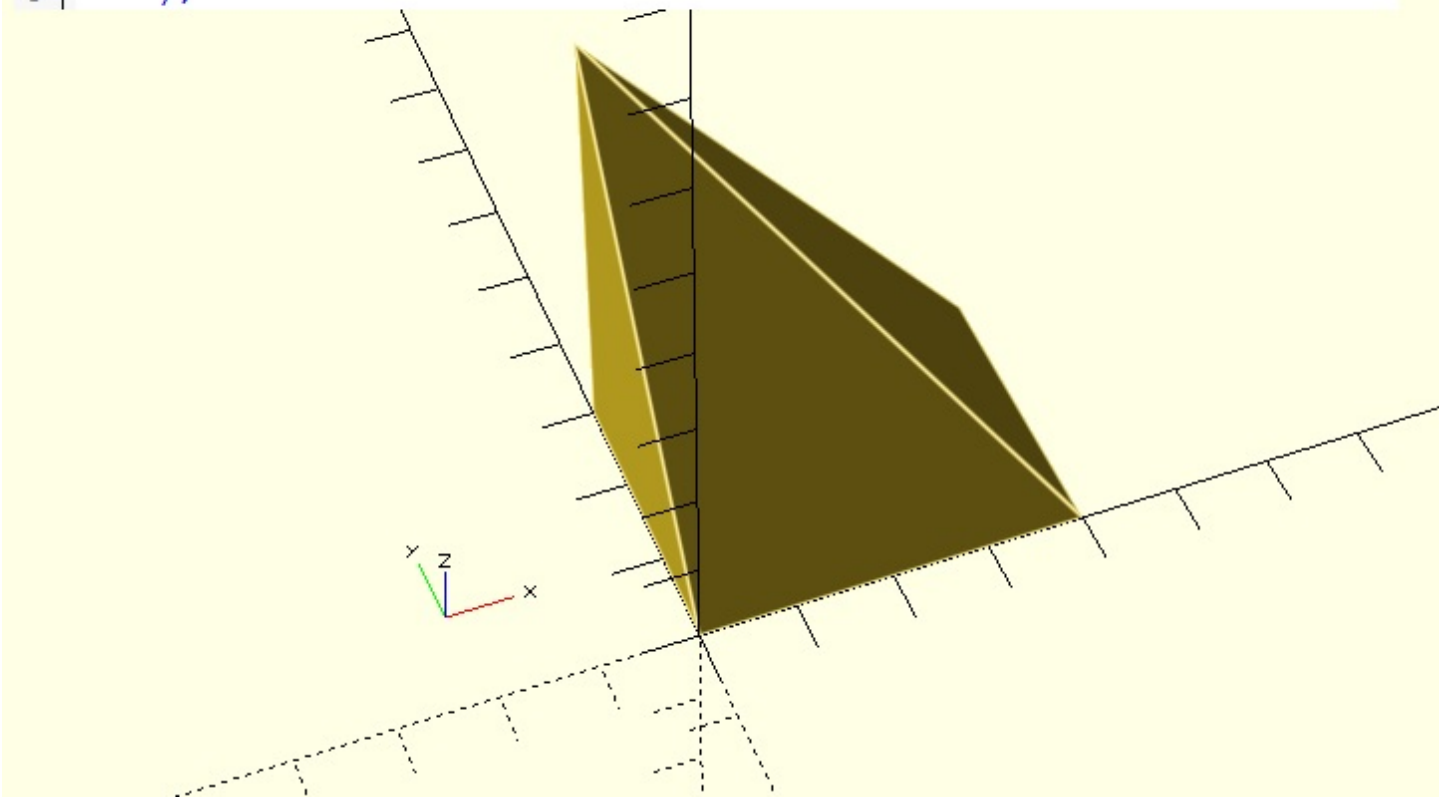
Exemple

```
1 polyhedron(
2 points=[[0,0,0],[40,0,0],[40,30,0],[0,30,0],[0,30,50]],
3 faces=[ [0,1,2,3],[0,1,4],[1,2,4],[2,3,4],[3,0,4] ]
4 );
```

produit

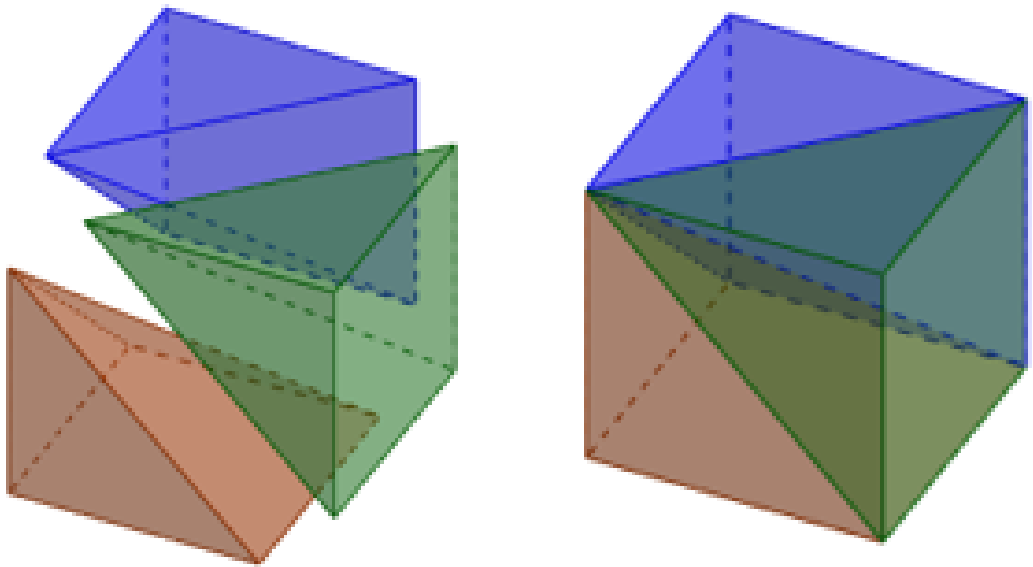
polyhedron() ;

```
polyhedron(
    points = [ [0,0,0],[40,0,0],[40,30,0],[0,30,0],[0,30,50] ],
    faces = [ [0,1,2,3],[0,1,4],[1,2,4],[2,3,4],[3,0,4] ]
    );
```



Activité 2

Trois pyramides pour un cube.



Lien vers animation geogebra

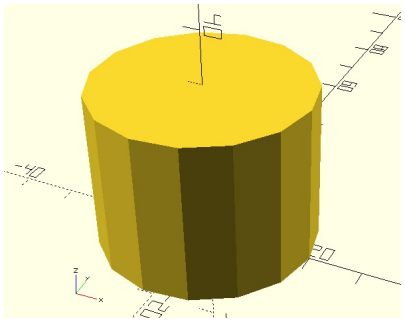
les solides prédéfinis : cylinder()

cylinder(h,r,center,$fn)

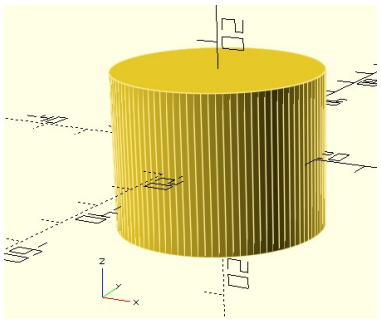
- r : rayon de la base du cylindre
- h : hauteur du cylindre
- center : optionnel, *true* ou *false* (par défaut)
- $fn : nombre de subdivisions pour les faces latérales (lissage), Plus précisément, on obtient un polygone de $fn côtés.
- cylinder(h,r1,r2,center,$fn) permet d'obtenir un cône (éventuellement tronqué, r1 rayon de la base inférieur, r2 rayon de la base supérieure)

cylinder() : exemples

```
1 cylinder(h=30,r=20,$fn=15);
```

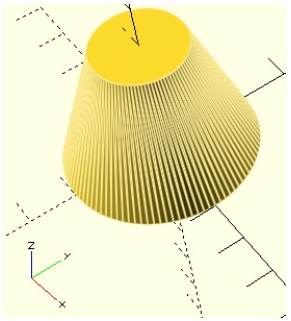


```
1 cylinder(h=30,r=20,center = true,$fn=100);
```

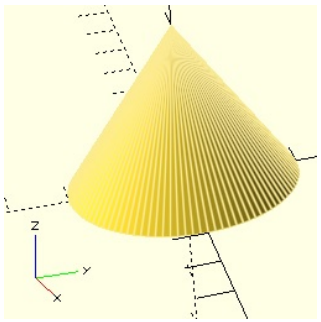


cylinder() : exemples - cônes

```
1 cylinder(h=30,r1=20,r2=10,$fn=100);
```



```
1 cylinder(h=30,r1=20,r2=0,$fn=100);
```



les solides prédéfinis : cube()

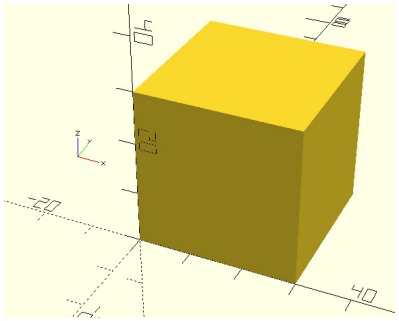
cube([x, y, z],center)

Pour construire un cube ou un pavé droit.

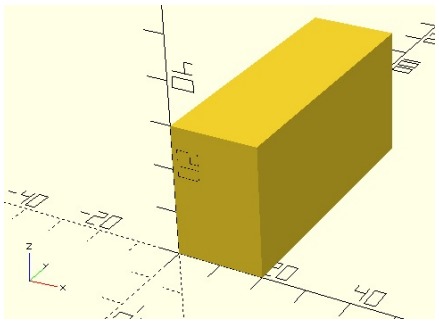
- $[x, y, z]$: dimensions du pavé droit
- cube(x) : permet d'obtenir un cube de côté x
- center : optionnel, *true* ou *false* (par défaut)

cube() : exemples

```
1 cube(30);
```



```
1 cube([20,60,30]);
```



les solides prédéfinis : sphere()

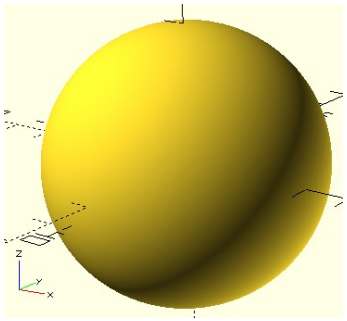
sphere(r,$fn)

Pour construire une boule.

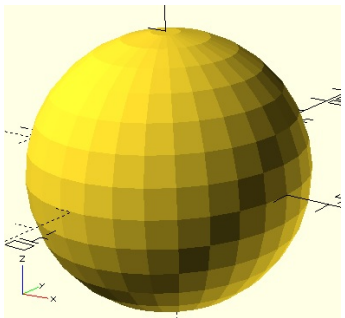
- r : rayon, on peut le remplacer par d pour le diamètre.
- $fn : optionnel, nombre de subdivisions (lissage)

sphere() : exemples

```
1 sphere(30,$fn = 500);
```



```
1 sphere(d=60,$fn = 30);
```



les fonctions (opérateurs) prédéfinies : translate()

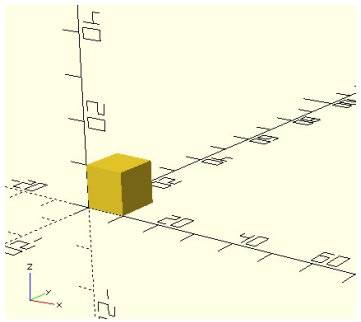
translate($[x, y, z]$) {...}

Pour translater un solide.

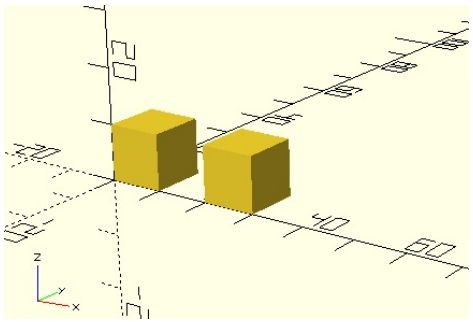
- $[x, y, z]$: vecteur de translation
- translate($[x, y]$) revient à effectuer translate($[x, y, 0]$)
- Dans les {} on place l'objet à translater, elles sont facultatives si un seul objet.

translate() : exemples

```
1 cube(10);
```

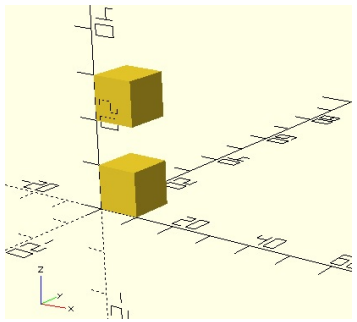


```
1 cube(10);
2 translate([20,0,0]) {cube(10);}
3 //ou translate([20,0,0]) cube(10);
```

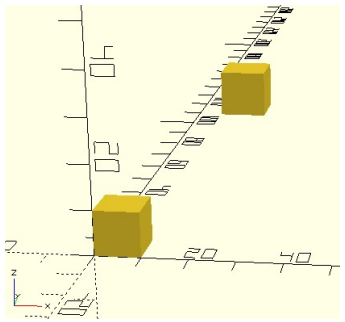


translate() : exemples

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



```
1 cube(10);
2 translate([20,40,20]) {cube(10);}
```



les fonctions (opérateurs) prédéfinies : rotate()

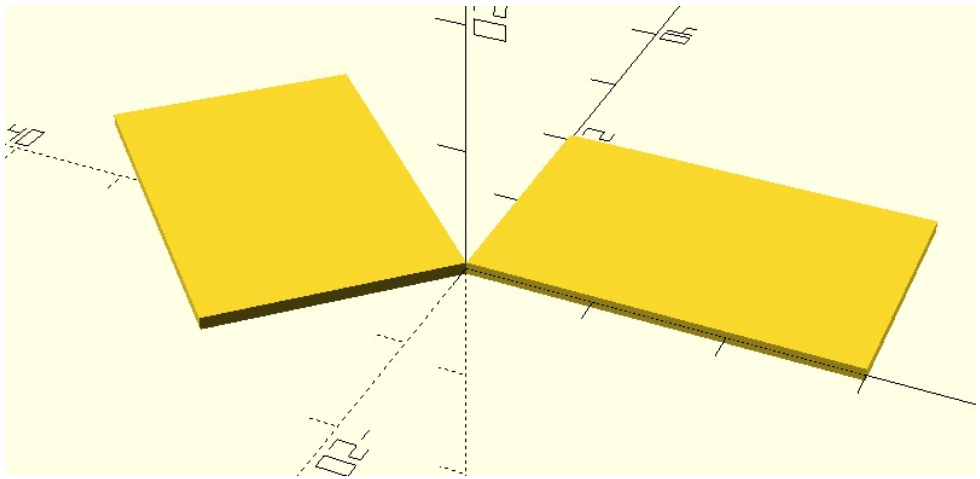
rotate(a, [x, y, z]) {...}

Pour effectuer une rotation du solide de a degrés autour de l'axe dirigé par le vecteur de coordonnées $[x, y, y]$, dans le sens direct (antihoraire)

- $[x, y, z]$: vecteur directeur de l'axe de rotation
- rotate(a) revient à effectuer rotate(a,$[0, 0, z]$)
- Dans les {} on place l'objet sur lequel appliquer la rotation, elles sont facultatives si un seul objet.

rotate() : exemples

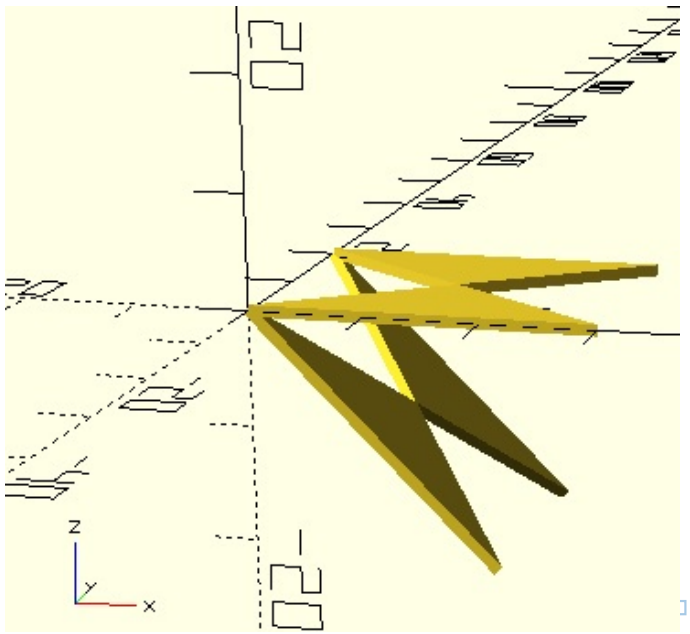
```
1 polygon([[0,0],[30,0],[30,20],[0,20]]);
2 rotate(135) {polygon([[0,0],[30,0],[30,20],[0,20]]);}
```



rotate() : exemples

Les accolades sont facultatives lorsqu'elles ne contiennent qu'un objet.

```
1 points = [[0,0],[30,0],[30,20],[0,20]];
2 chemin = [[0,2,3,1]];
3 polygon(points,chemin);
4 rotate(45,[0,1,0]) polygon(points,chemin);
```



les fonctions (opérateurs) prédéfinies : mirror()

mirror([x, y, z]) {. . .}

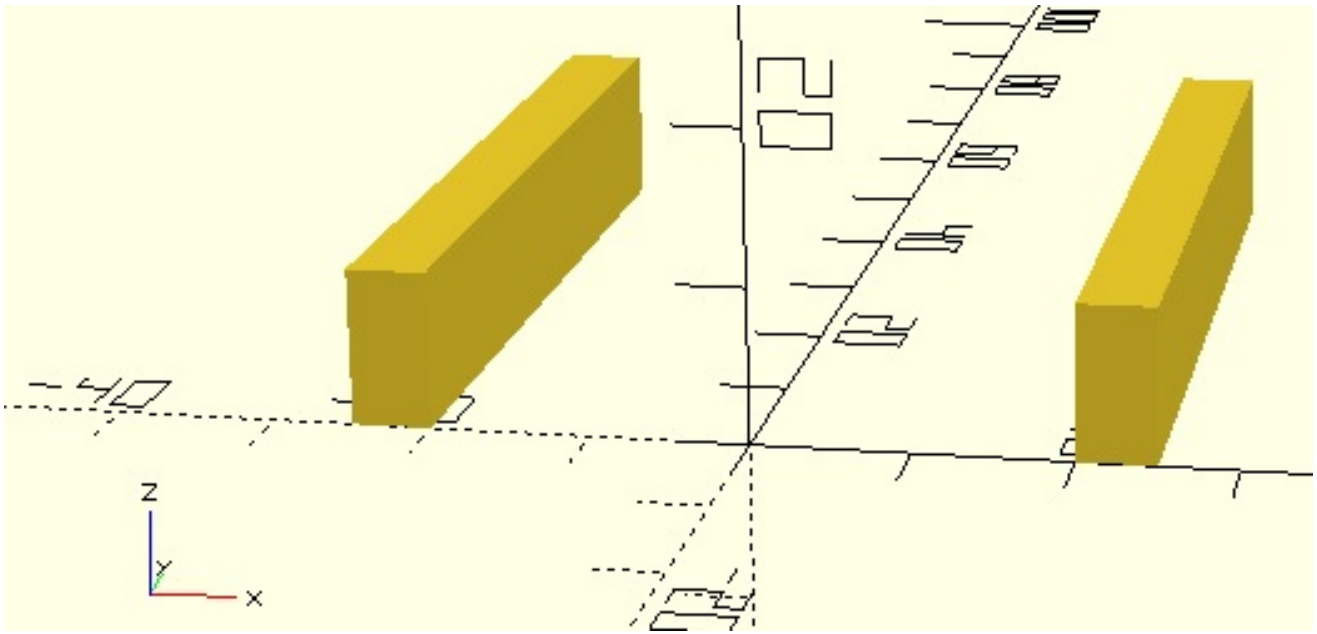
Pour effectuer une symétrie par rapport à un plan de vecteur normal le vecteur de coordonnées $[x, y, y]$

- $[x, y, z]$: vecteur normal du plan de la symétrie
- Dans les {} on place l'objet sur lequel appliquer la symétrie

mirror() : exemples

Les accolades sont facultatives lorsqu'elles ne contiennent qu'un objet.

```
1 translate([20,0,0]) cube([5,50,10]);
2 mirror([1,0,0]) {translate([20,0,0]) cube([5,50,10]);}
```

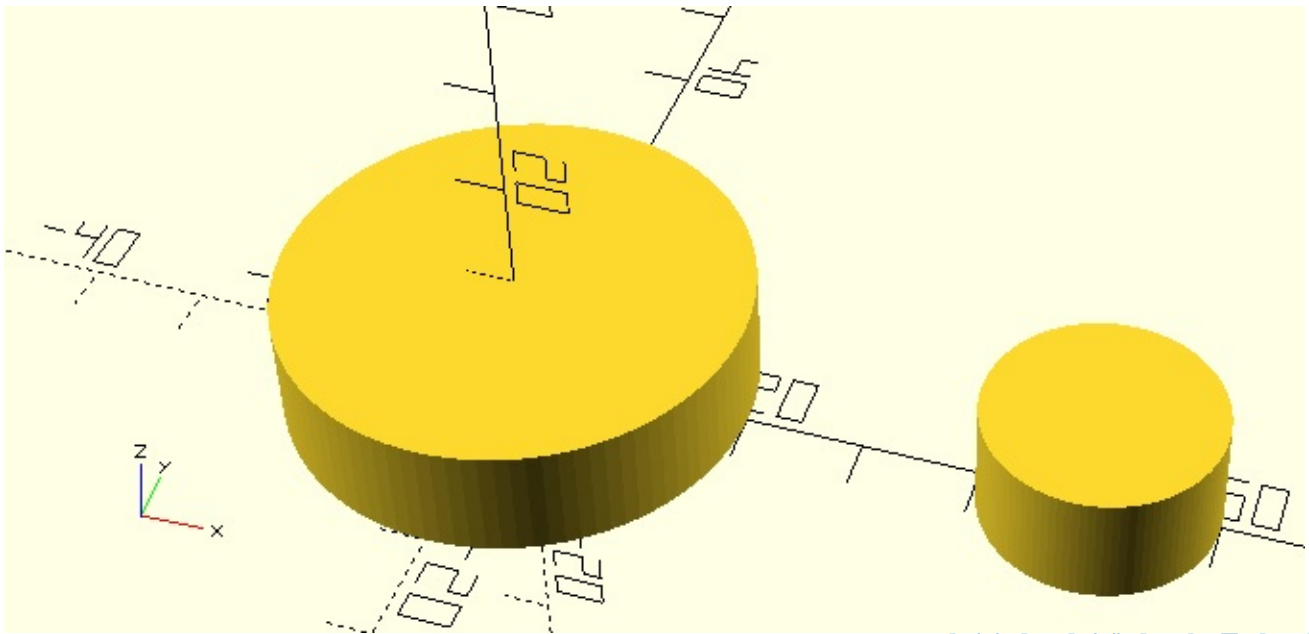


les fonctions (opérateurs) prédéfinies : hull()

hull() . . .

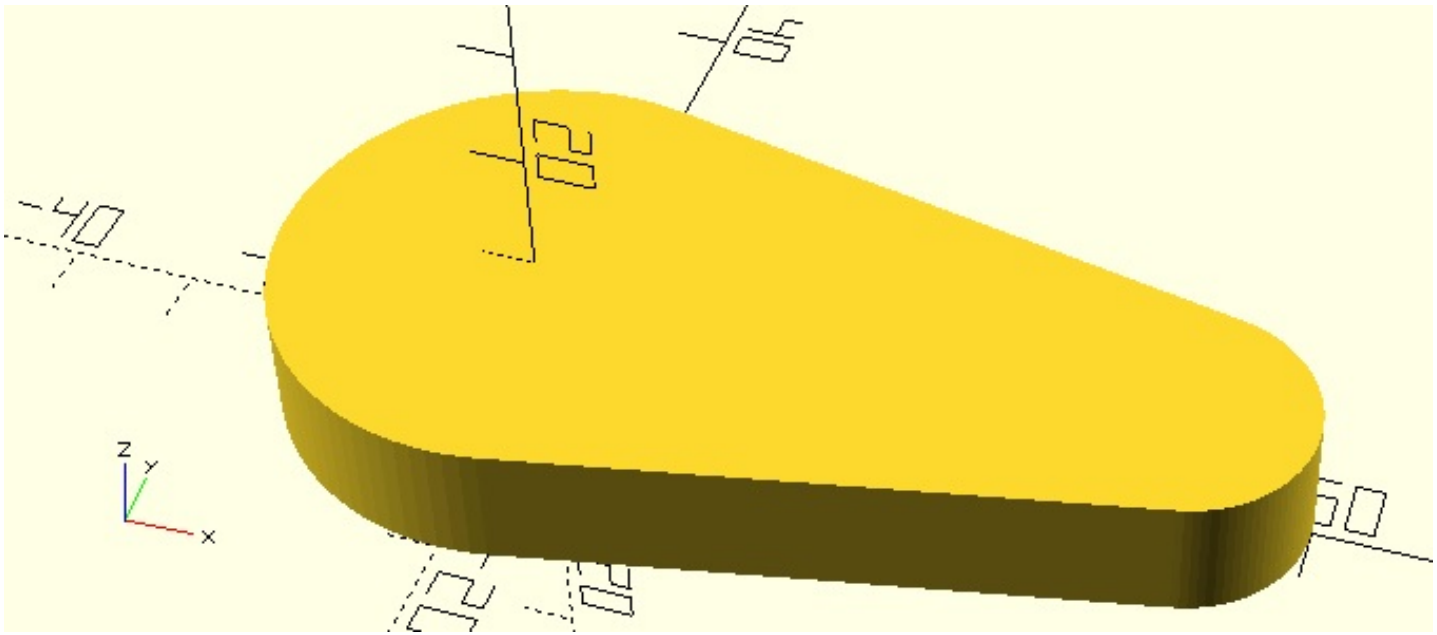
Pour obtenir l'enveloppe convexe de l'objet décrit entre les {}

```
1 cylinder(h=10,r=20,$fn=100);
2 translate([50,0,0]) cylinder(h=10,r=10,$fn=100);
```



hull() : exemple

```
1 hull(){
2   cylinder(h=10,r=20,$fn=100);
3   translate([50,0,0]) cylinder(r=10,h=10,$fn=100);
4 }
```

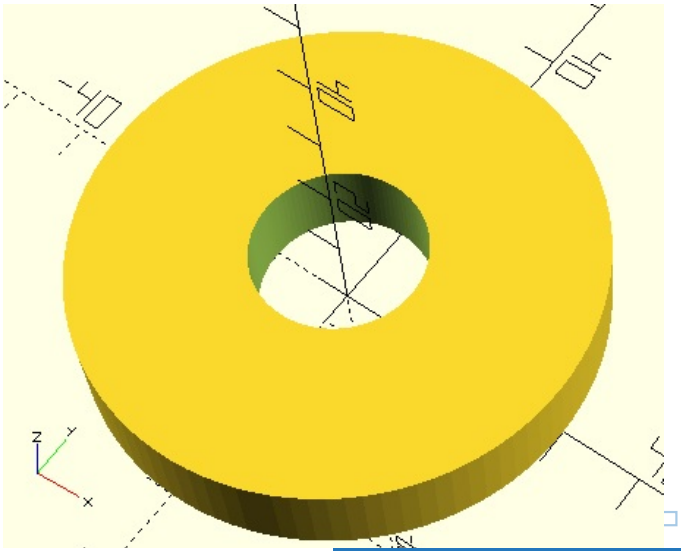


les fonctions (opérateurs) prédéfinies : difference()

difference() {. . .}

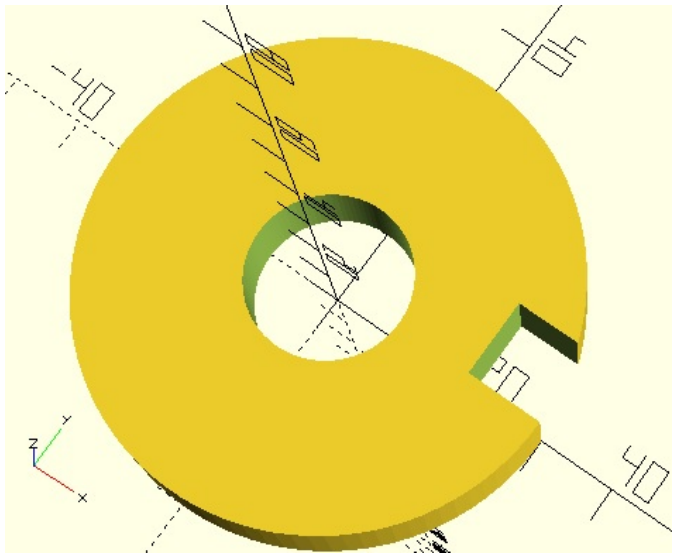
Pour soustraire du premier objet les suivants.

```
1 difference(){
2   cylinder(h=10,r=30,$fn=100);
3   cylinder(h=10,r=10,$fn=100);
4 }
```



difference() : exemple

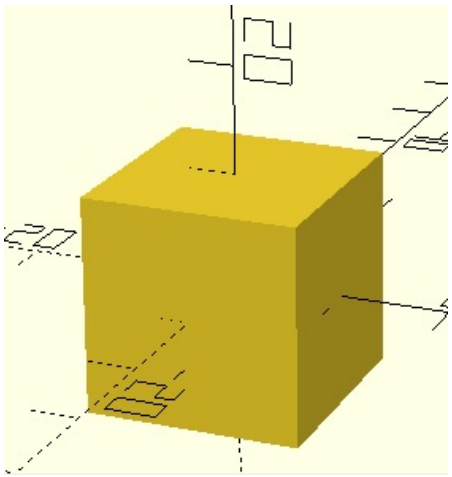
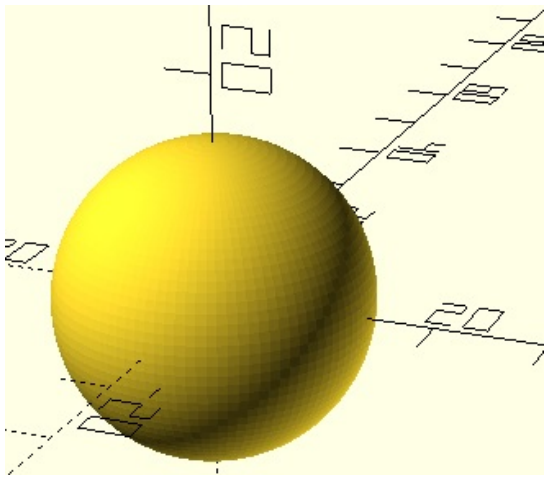
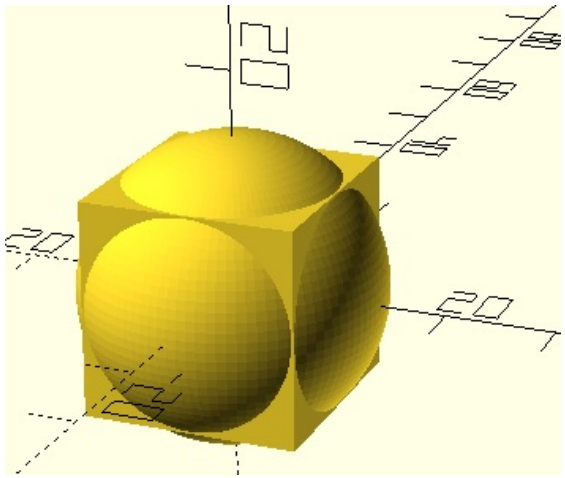
```
1 difference(){
2   cylinder(h=10,r=30,$fn=100);
3   cylinder(h=10,r=10,$fn=100);
4   translate([20,0,0]) cube(10);
5 }
```



les fonctions (opérateurs) prédéfinies : intersection()

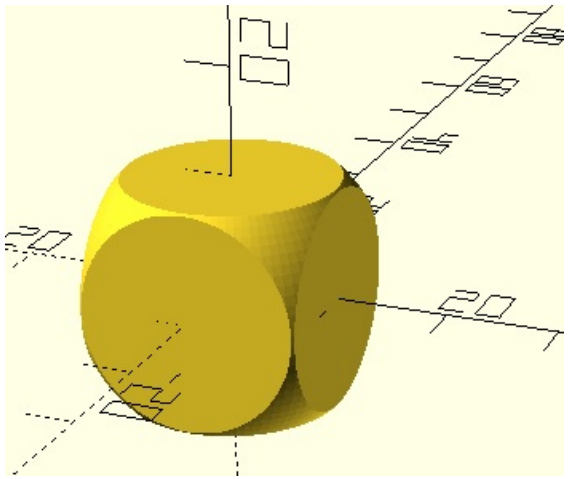
intersection() {...}

Pour construire l'intersection entre un premier objet et les suivants.

intersection() : exemple

```
1 intersection(){
2   cube(20,center=true);
3   //Pour le rayon de la sphère,
4   //prendre 70% du côté du cube
5   sphere(14,$fn=100);
6 }
```



les fonctions (opérateurs) prédéfinies : union()

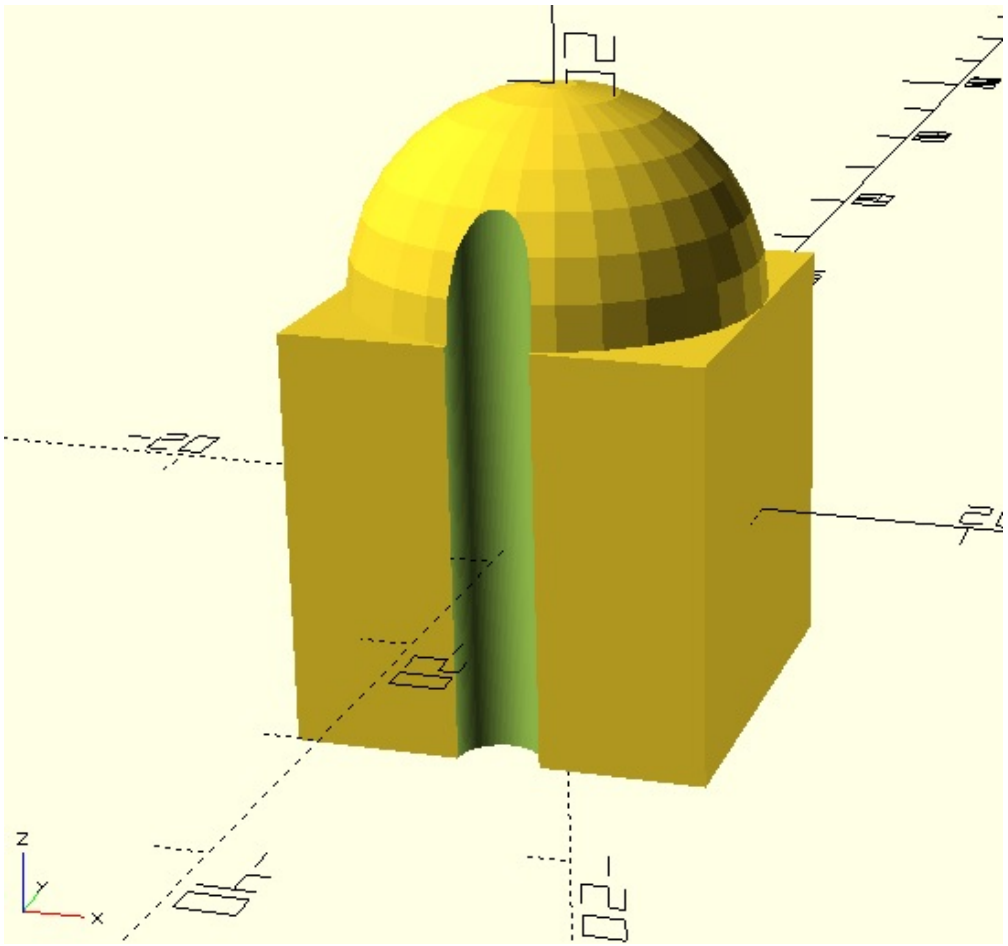
union() {...}

Pour fusionner plusieurs objets.

- Openscad fusionne par défaut les objets
- A utiliser avec un autre opérateur, dans ce cas très utile

```
1 difference(){
2   union(){
3     cube(20,center=true);
4     translate([0,0,10]) sphere(10);
5   }
6   translate([0,-10,-15]) cylinder(h=40,r=2,$fn=100);
7 }
```

union() : exemple



les fonctions (opérateurs) prédéfinies : rotate_extrude()

rotate_extrude(angle,$fn) {...}

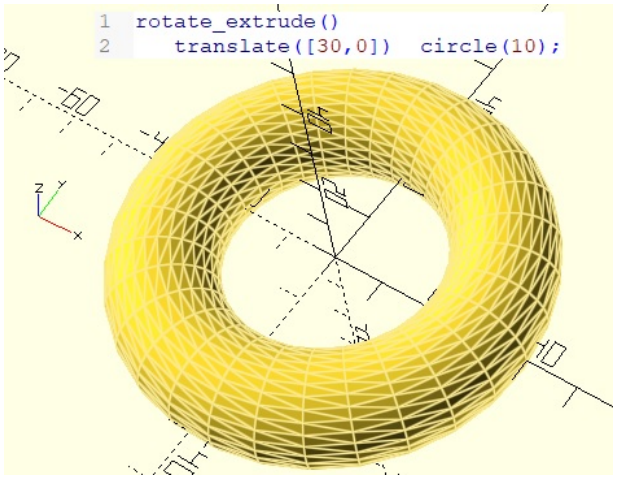
- angle : optionnel, en degré, angle de rotation autour de l'axe (Oz)
- $fn : optionnel,nombre de subdivisions (lissage)
- Attention, la forme de départ doit être incluse dans le plan (x0y)

Exemples : rotate_extrude()

Attention, la forme de départ doit être incluse dans le plan (x0y)

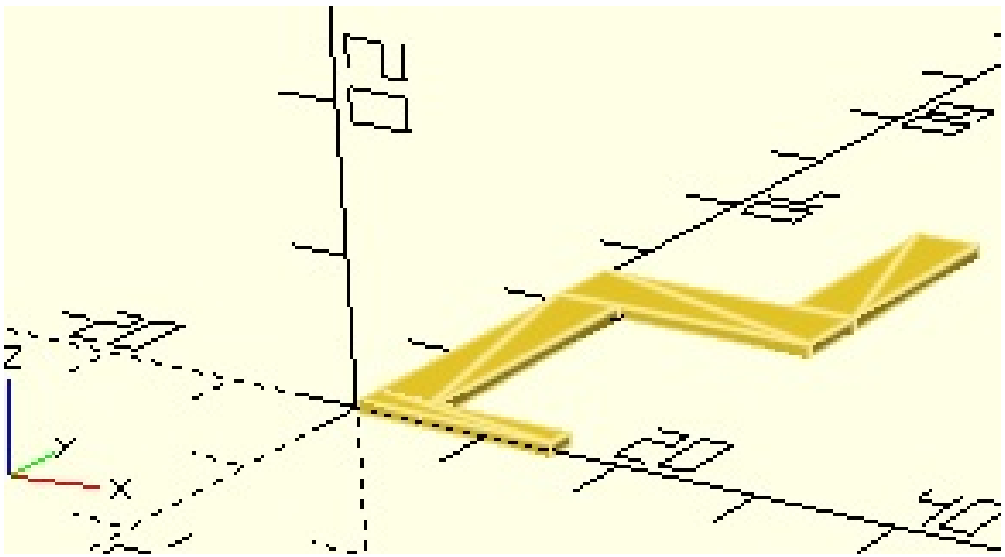



```
1 rotate_extrude()
2   translate([30,0])  circle(10);
```

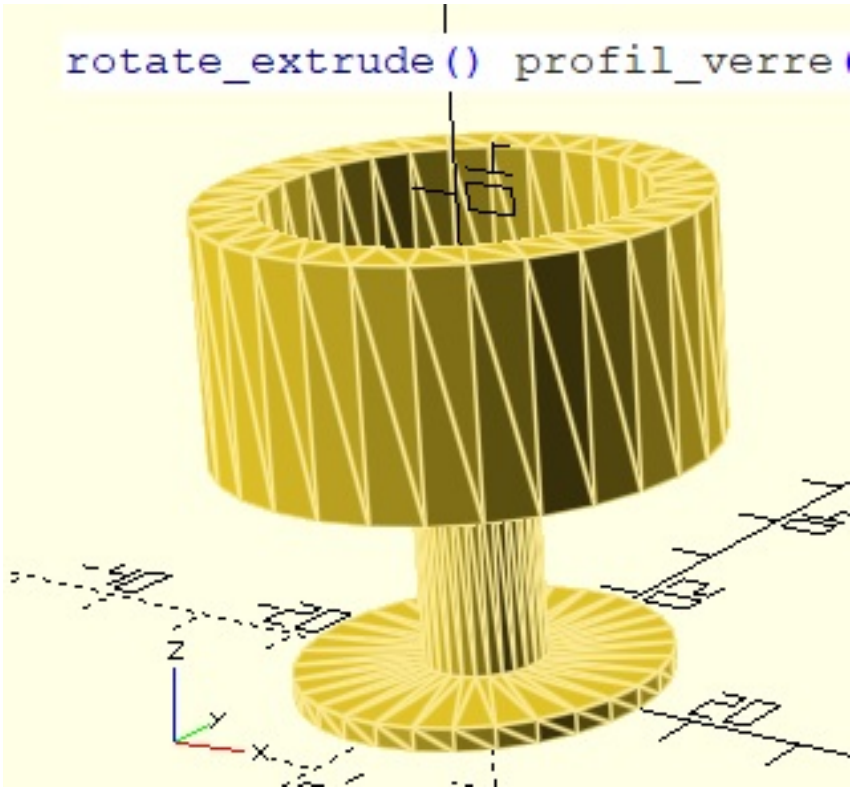


Exemples : rotate_extrude()

```
1 module profil_verre(){
2   square([5,20]);
3   square([15,2]);
4   translate([0,20]) square([20,5]);
5   translate([15,25]) square([5,15]);
6 }
7 profil_verre();
```



Exemples : rotate_extrude()



Créer une fonction : module nomdumodule()

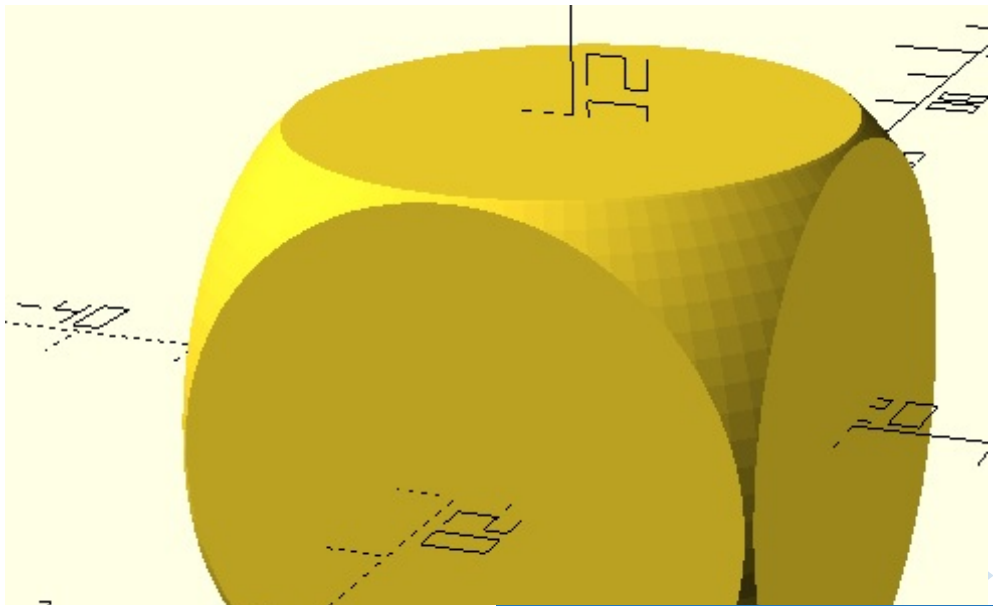
module nomdumodule(a,b,c) {. . .}

Pour créer un bout de code destiné être utilisé plusieurs fois

- Les paramètres a,b,c. . .sont optionnels
- Dans les {} on place les instructions

module nomdumodule() : exemple

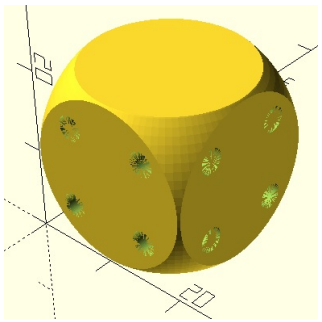
```
1 module mon_de(cote){
2   intersection(){
3     cube(cote,center=true);
4     sphere(cote*0.70,$fn=100);
5   }
6 }
7 mon_de(40);
```

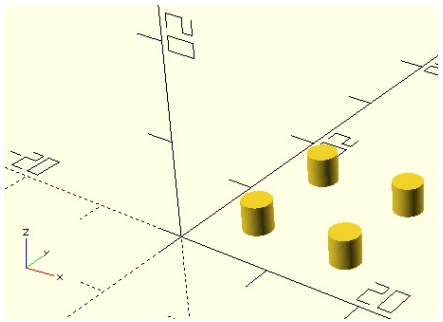


Activité 3

Construire le dé d'Efron [4,4,4,4,0,0] en utilisant le module défini ci-dessous et sur lequel on appliquera des rotations et translations. $fn=100 peut être écrit une seule fois en début de script.

```
1 module quatre() {
2   translate([5,15,0]) cylinder(d=3,h=3,$fn=100);
3   translate([15,15,0]) cylinder(d=3,h=3,$fn=100);
4   translate([5,5,0]) cylinder(d=3,h=3,$fn=100);
5   translate([15,5,0]) cylinder(d=3,h=3,$fn=100);
6 }
7 //quatre();
```





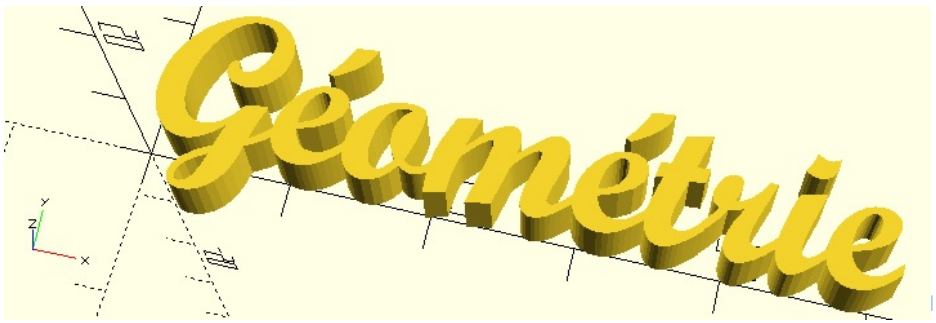
Ajouter du texte

text(montexte,size,font)

Pour créer du texte.

- montexte est une chaîne de caractère entre ""
- size : la taille de la police
- font : la police utilisée, à choisir dans : Aide⇒Liste des polices

```
1 linear_extrude(5)
2   text("Géométrie",font="Script MT Bold:style=Regular",
3   size = 10);
```



Activité 4

Deux exercices proposés :

- Construire le carré "inconnue x"

  ou

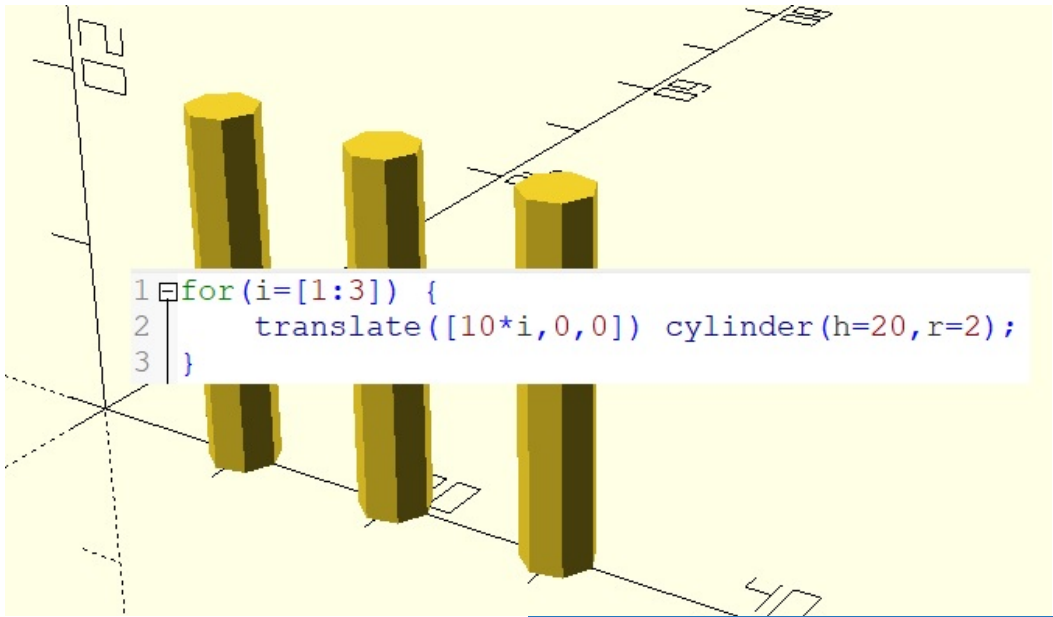
- Construire les pièces de l'identité remarquable $(a+b)^2$

La boucle Pour

for(i=[a :b]) {...}

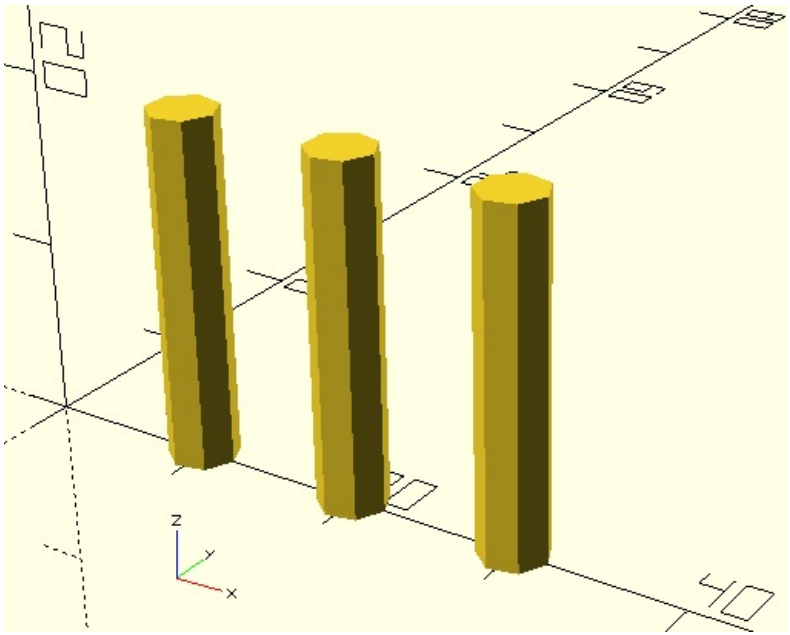
Boucle pour, la variable i prend toutes les valeurs entières de a à b inclus

- [*a* : *b*] peut être remplacé par [valeur départ :pas :valeur fin]
- [*a* : *b*] peut être remplacé par une liste de nombres [e,f,g,h,i,j]



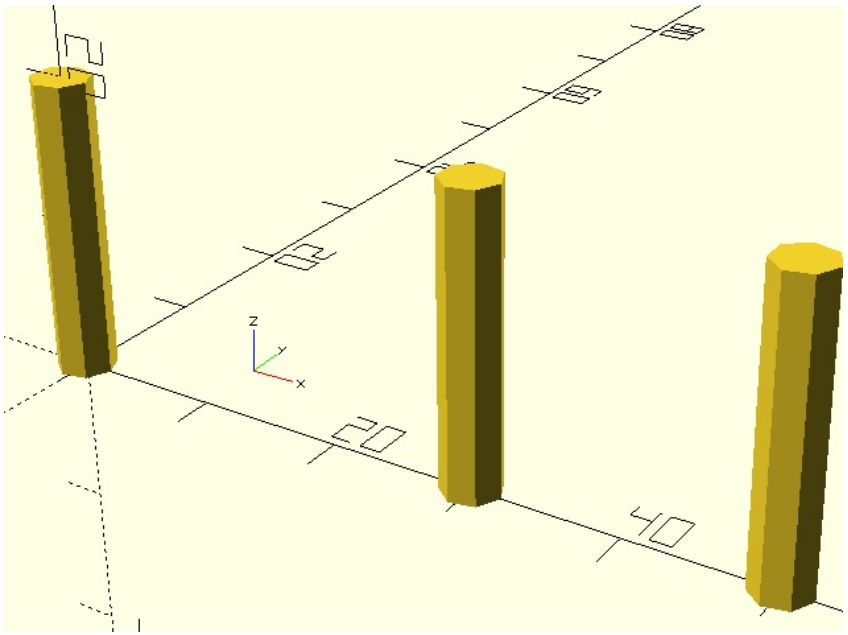
La boucle Pour : exemples

```
1 for(i=[10:10:30]) {
2   translate([i,0,0]) cylinder(h=20,r=2);
3 }
```



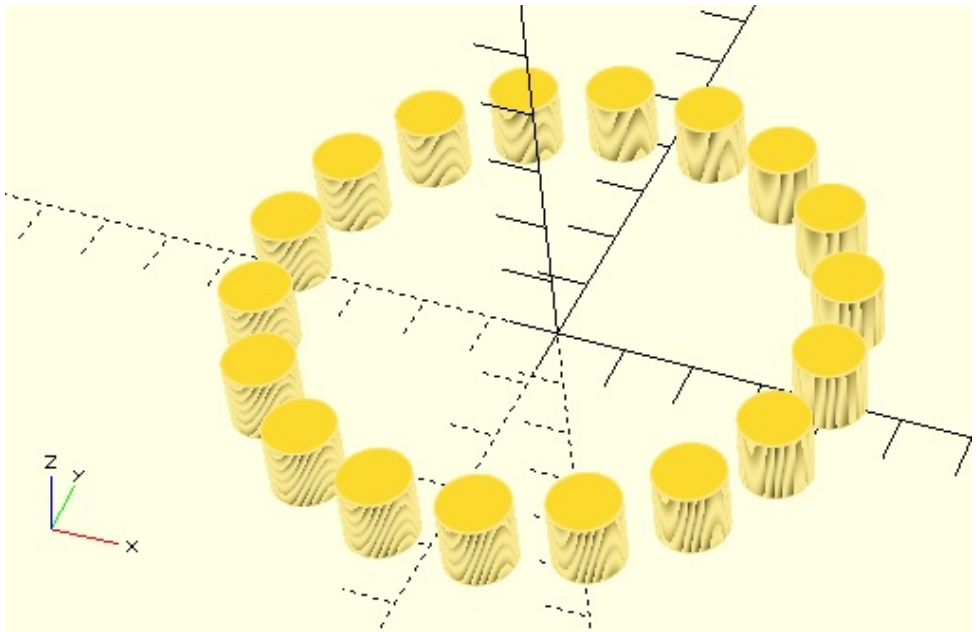
La boucle Pour : exemples

```
1 for(i=[0,30,50]) {
2   translate([i,0,0]) cylinder(h=20,r=2);
3 }
```



La boucle Pour : exemples

```
1 $fn = 100;
2 for(angle=[0:20:360]) {
3   rotate([0,0,angle])
4     translate ([40,0,0]) cylinder(h=10,r=5);
5 }
```



Autres : if (condition) {...} et intersection_for

if (condition) {...}

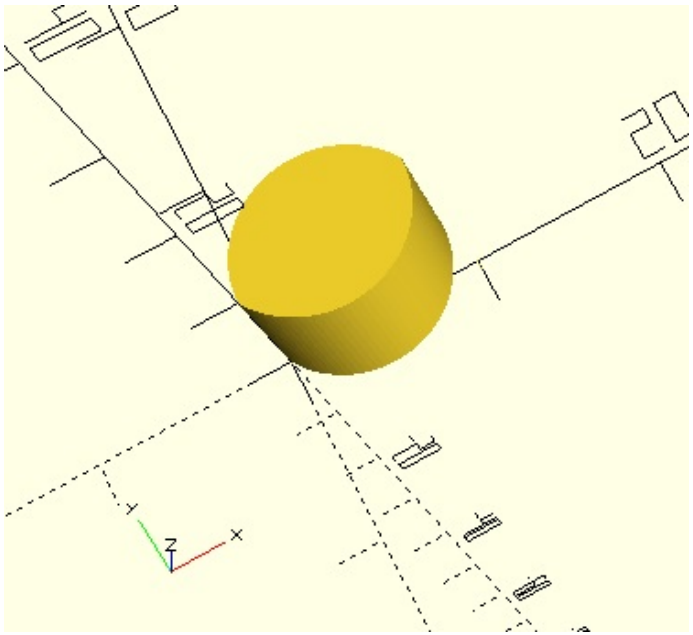
- Si...alors... «classique»
- Si...alors...sinon s'écrit : if (condition) {...} else {...}

intersection_for(i=[a :b]) {...}

- Pour créer une intersection/chevauchement de tous les objets générés.
- La variable i s'initialise comme pour la boucle for

La boucle intersection_for : exemple

```
1 $fn = 100;
2 intersection_for(angle=[0:10:30]) {
3   rotate([0,0,angle])
4    translate ([5,0,0]) cylinder(h=10,r=5);
5 }
```



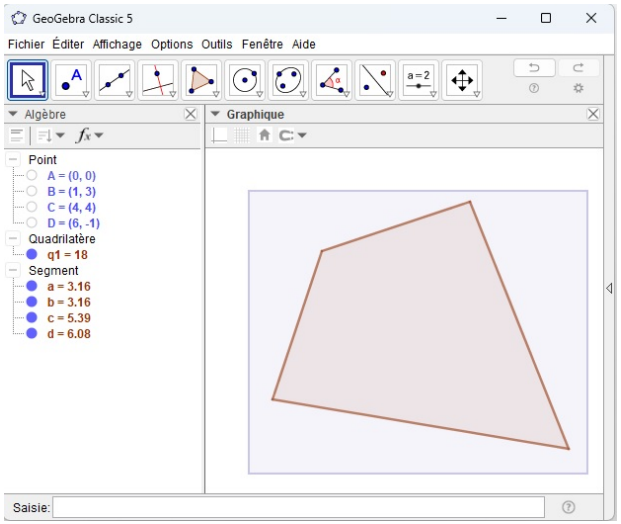
Import d'image SVG

- Créer une figure dans Geogebra, puis enlever axes et grille
- «Exporter Graphique en tant qu'image»au format png en 300 dpi,
- Utiliser un convertisseur png ⇒ svg, par exemple en ligne convertio
- Dans OpenScad, enregistrer le fichier de travail dans le même répertoire que l'image (ou détailler le chemin d'accès) et écrire par exemple le script ci-dessous :

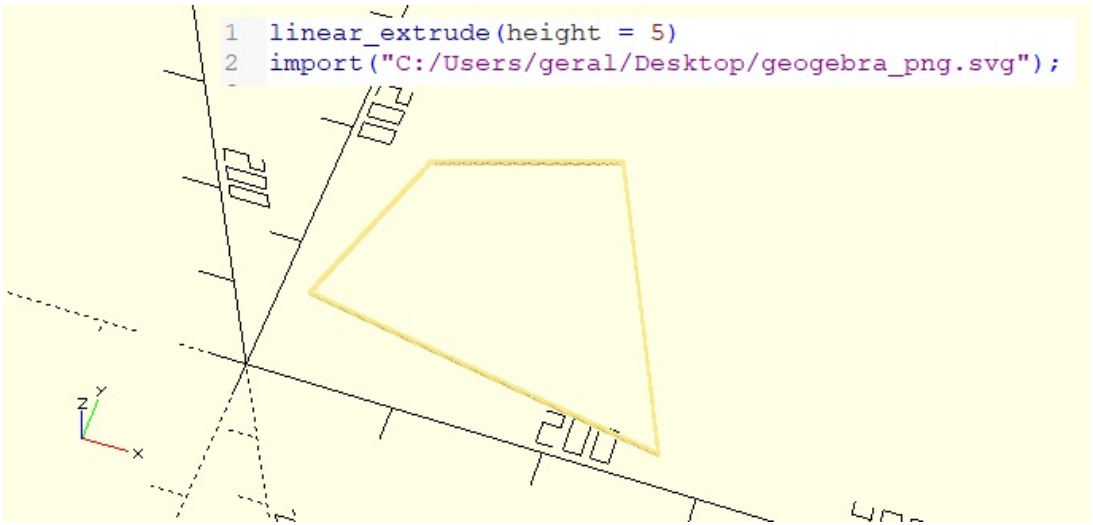
```
1 linear_extrude(height = 5)
2 import("C:/Users/geral/Desktop/geogebra_png.svg");
```

- Une procédure plus détaillée (et plus complexe) est détaillé dans le wiki

Import d'image SVG : illustrations




```
1  linear_extrude(height = 5)
2  import("C:/Users/geral/Desktop/geogebra_png.svg");
```



Fonction mathématiques

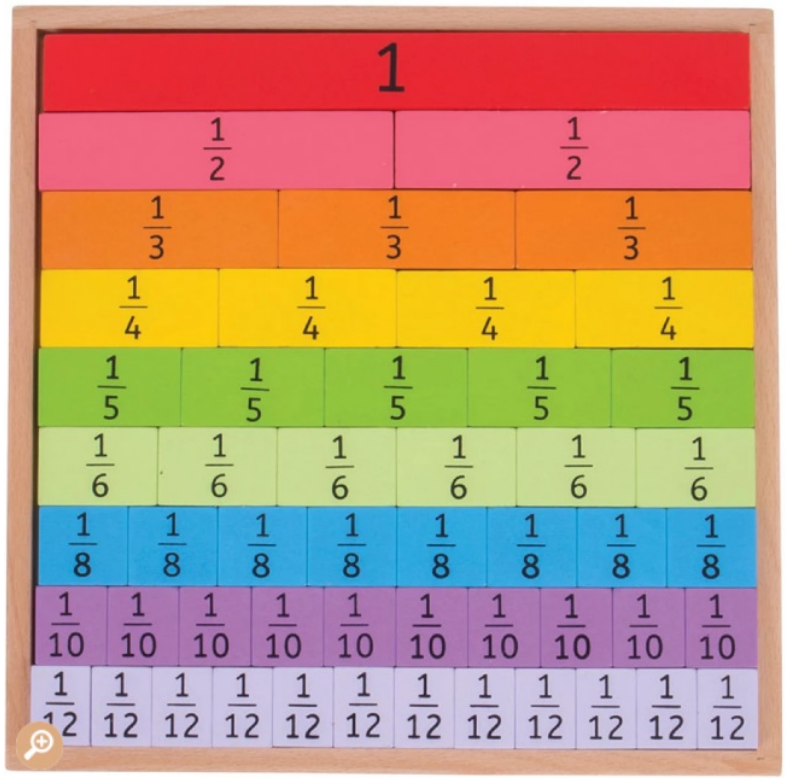
OpenSCAD permet l'utilisation de nombreuses fonctions mathématiques, décrites sur le
wiki des fonctions mathématiques (en anglais)

Autres idées d'objets pour la classe de mathématiques

- Les portes (= gobelets) de Monty-Hall
- Des objets pour les unités d'aires, de volumes...(100 carrés de 1 cm qui remplissent un carré de 1 dm de côté...)
- Un dé truqué
  Laboratoire de Mathématiques du Lycée Zola de Rennes

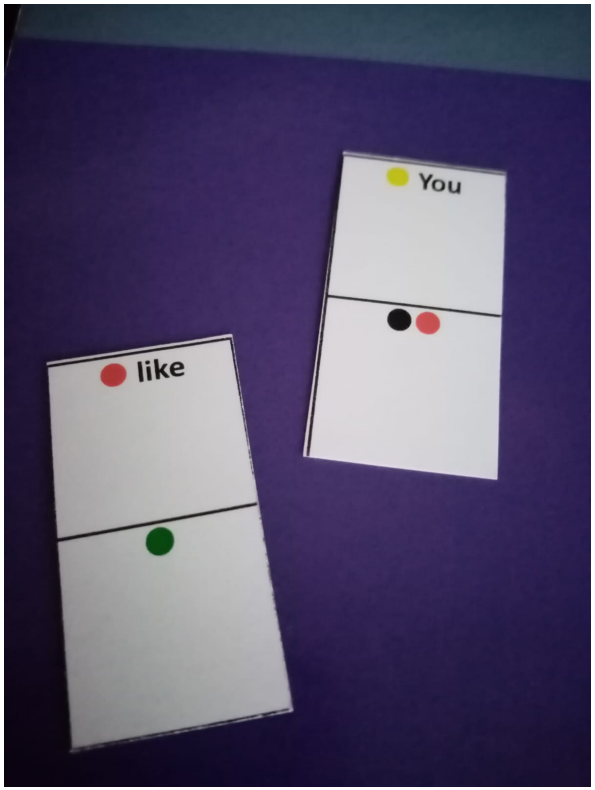
Autres idées d'objets pour la classe de mathématiques

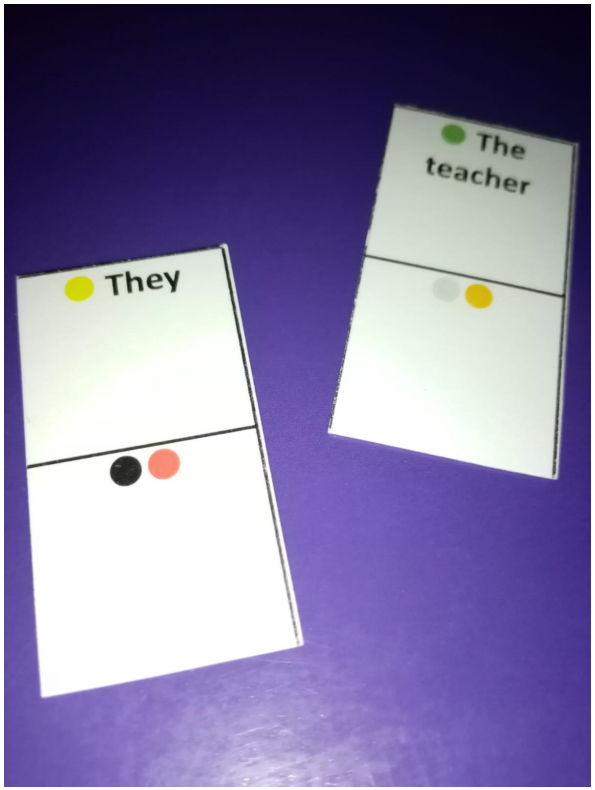
- Des objets pour manipuler les fractions (camemberts...)
- Fractions en bandes



Autres idées d'objets pour les langues

- Les dominos linguistiques





Sites de référence

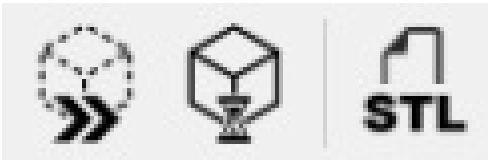
- wikibook (en anglais)
- un autre wikibook (en français)
- wiki des fonctions mathématiques (en anglais)
- un livre :
  Géométrie dans l'espace et IMPRESSION 3D, J. JACQUET



- Enfin, de nombreux tutoriels sont disponibles en ligne.

Impression : Enregistrer l'objet au format stl

- Dans un premier temps, il faut enregistrer le fichier au format stl :
  - OpenSCAD : en appuyant successivement sur les touches



  - FreeCad : en exportant l'objet au format stl
- Dans un second temps, il faut ouvrir le fichier au format stl en utilisant le logiciel Ultimaker Cura (trancheur)

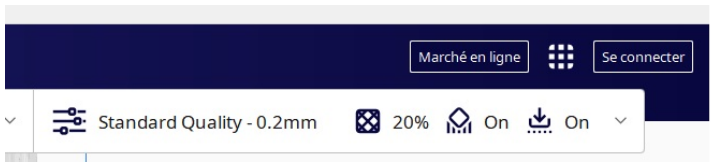
Impression : Ultimaker Cura



- logiciel libre de découpe (tranches d'impression)
- performant, complet et simple d'utilisation
- lien pour téléchargement : https://ultimaker.com/software/ultimaker-cura/
- au démarrage (ou plus tard) bien choisir votre imprimante parmi celles proposées, ou une imprimante compatible ou entrer les paramètres proposés par le fabricant
- choisr le bon filament (souvent PLA) et le bon diamètre de la buse (nozzle, souvent 0.4mm)

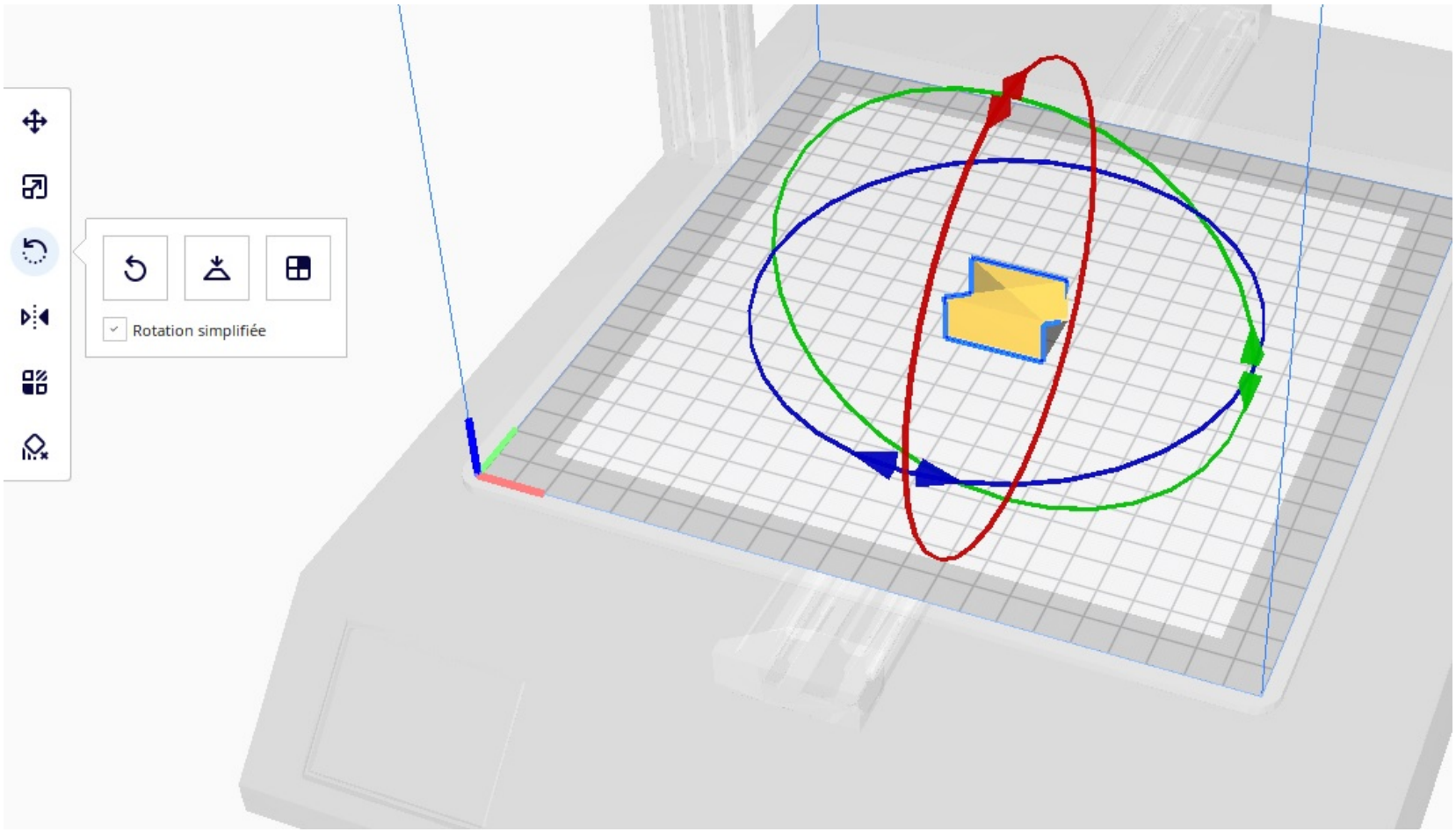
Impression : Ultimaker Cura

Dérouler le menu des paramètres d'impression à droite

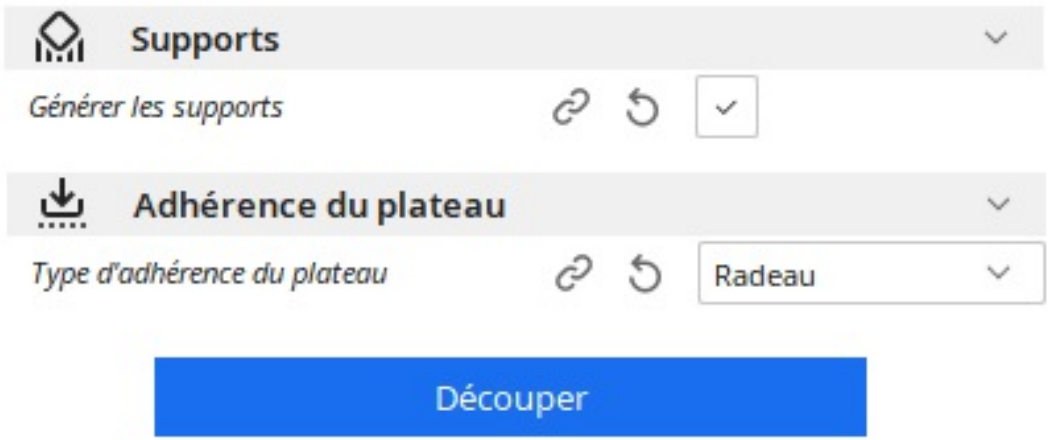


- Régler la température d'impression : souvent 200°C ou plus
- Régler la température du plateau pour une bonne adhérence : souvent 60°C
- A priori, cocher la case "générer les supports"
- Choisir "Radeau" pour le "Type d'adhérence du plateau"
- Eventuellement utiliser le menu à gauche après avoir cliqué sur la pièce

Impression : Ultimaker Cura



Impression : Ultimaker Cura



- Cliquer sur aperçu
- Maintenir le clic droit et bouger la souris afin de vérifier la pièce sous différentes vues
- Il ne reste plus qu'à transférer le fichier .gcode obtenu sur une carte sd et à lancer la procédure d'impression sur l'imprimante.

Retour

Impression : Types de filaments

Attention, il faut vérifier les types de filaments acceptés par l'imprimante 3D

- PLA : Bonne résistance mécanique (flexion, traction), mauvaise résistance thermique au-delà de 50°C
- ABS : Bonne résistance mécanique (impact, traction), meilleure résistance thermique que le PLA, moins bonne adhérence (couches, plateau) que le PLA.
- PET-G : Filament intermédiaire entre PLA et ABS
- TPU : Filament flexible, plus rarement accepté par l'imprimante
- Liste non exhaustive

Conseils

- Attention aux trous
- Attention aux supports
- Bien choisir la base de l'objet qui va reposer sur le plateau