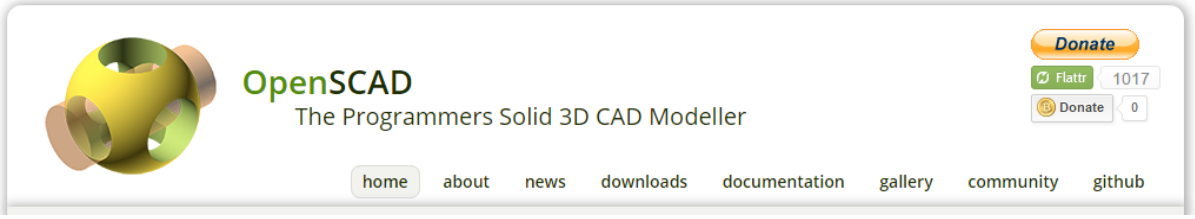
UNIVERSITÉ TOULOUSE III PAUL SABATIER

Atelier openSCAD

Avril 2017 – v. 1.2

1. introduction

OpenSCAD (**http://openscad.org**) est un logiciel libre pour la modélisation 3D disponible pour Linux, Windows et MacOS. Ce logiciel permet de générer des figures au format STL, format largement utilisé pour l'impression 3D.



2. installer OpenSCAD

La dernière version disponible date de mars 2015. Les modalités d'installation sont données sur la site web suivant les systèmes d'exploitation. Au démarrage (cf. Figure 1), propose des exemples afin de prendre en main l'outil ou de créer un nouvel objet 3D.

Figure 1 : écran de démarrage d'OpenSCAD

Quelques raccourcis sont utiles pour manipuler l'outil plus facilement.

Rafraîchir la vue en mode brouillon	*Touche F5* 
Compiler le rendu	*Touche F6* 
Commenter / décommenter le code	*Touches CTRL+D / SHIFT+CTRL+D*
Indenter/Annuler l'indentation	*Touches CTRL+I / SHIFT+CTRL+D*

3. mon premier objet

Le langage utilisé par OpenSCAD repose sur un ensemble de primitives (voir **http://www.openscad.org/datasheet** pour l'ensemble des mots-clés) et d'opérateurs de transformations mathématiques appliquées sur l'objet.

OpenSCAD permet donc de « programmer » ces objets 3D à l'instar de langages par blocs comme **blockscad3D** (**https://www.blockscad3d.com/editor**).

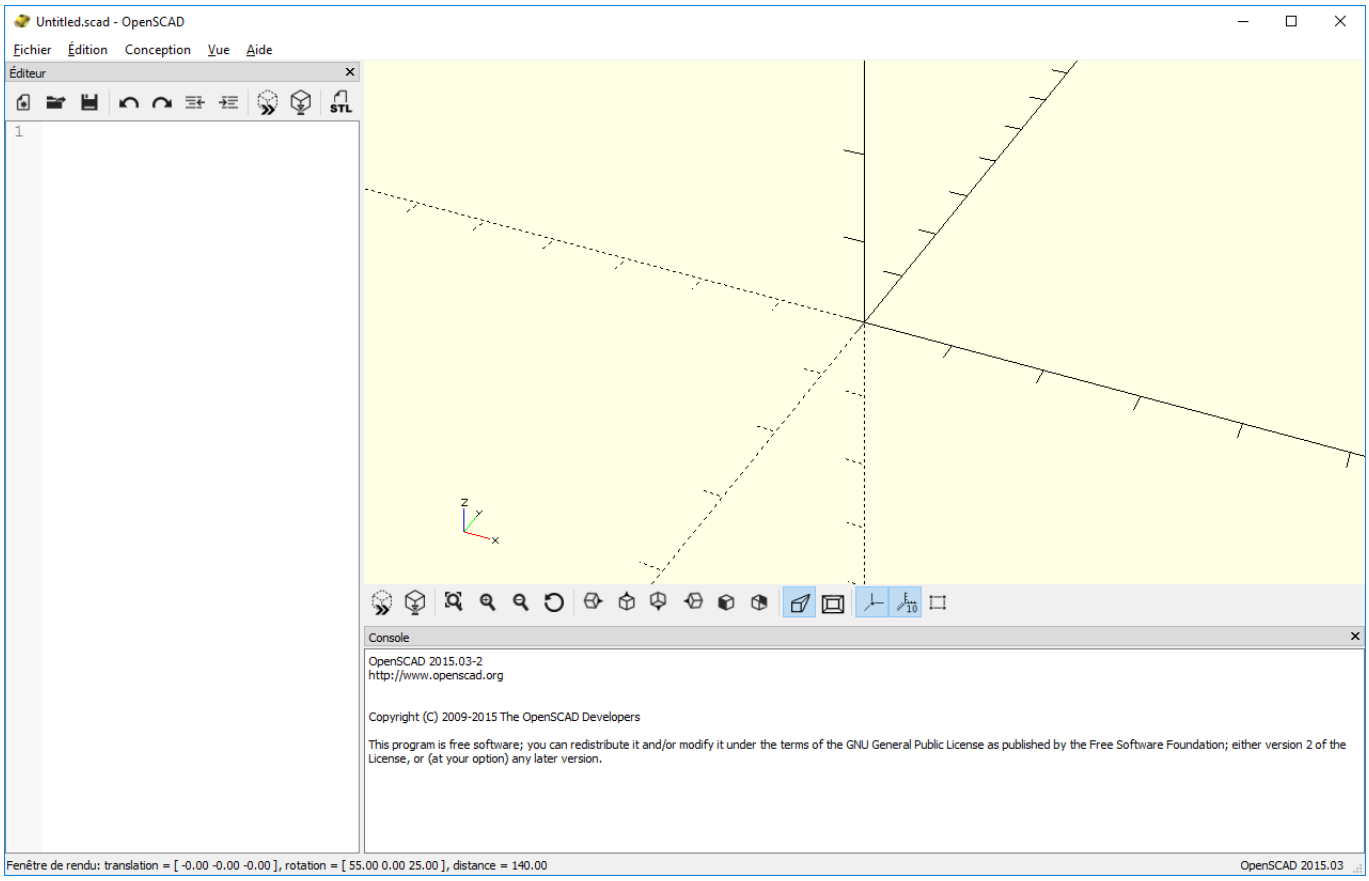


Figure 1 : Fenêtres du logiciel OpenSCAD

Concernant la fenêtre d'édition cf. Figure 1), la fenêtre de gauche permet de coder l'objet tandis que celle de droite permet de visualiser le résultat correspondant

Les principales primitives graphiques 3D sont ; *sphere*, *cube*, *cylinder* et *polyhedron*
D'autres primitives en 2D existent : *square*, *circle* et *polygon*

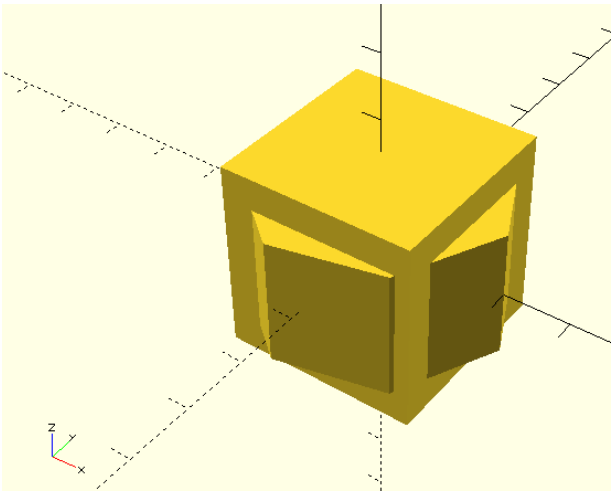
Les transformations applicables peuvent être t : *translate*, *rotate*, *scale*, *mirror*, etc …
Enfin, les opérations booléennes *d'union*, de *difference* ou *d'intersection* sont possibles.

Par exemple, nous allons construire un cube d'arête « **3** » (l'unité importe peu – l'unité par défaut est le mm) et un cylindre imbriqué dans le celui-ci en les centrant sur l'origine.

Les instructions :

```
cube (3, center = true) ;
cylinder (r=2 , h= 2 , center = true) ;
```

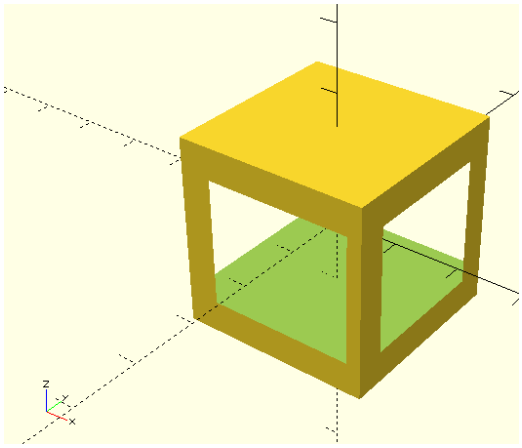
Le résultat produit sera :



Nota : le rendu du cylindre vous semble étrange ? C'est normal ! …

Si nous souhaitons obtenir la différence de ces deux primitives, il suffit d'appliquer l'opération *différence* sur ces deux instructions :

```
difference() {
      cube (3, center=true) ;
      cylinder (r=2 , h= 2 , center = true) ;
}
```



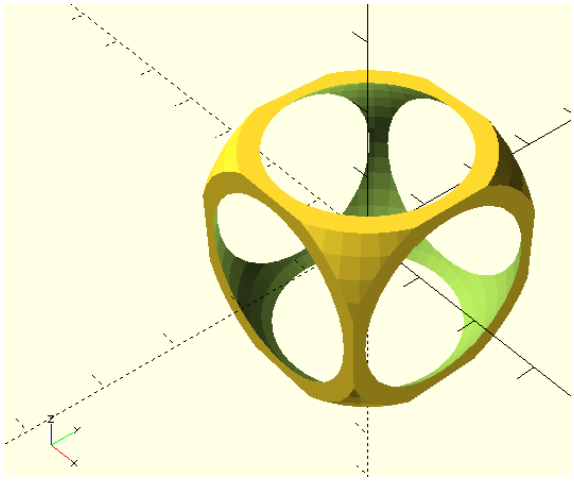
Le résultat en devient donc :

Les opérations peuvent bien évidemment être imbriquées (*Attention à l'ordre* !).
Le rendu peut être parfois assez long !

Nota : afin d'accélérer le rendu sur des cartes graphiques peu performantes, Aller dans le menu **Edition | Préférences**, onglet **Avancé** et cocher la case « **Forcer GoldFeather** »

Nota : Le résultat de la vue « *brouillon* » est parfois étrange, préférez la vue « rendu » 

Astuce : Afin d'affiner les figures tracées, il faut utiliser l'instruction : **`$fn = valeur`** (une valeur de 30 à 50 est recommandée)



Tracez maintenant l'objet suivant et exportez-le au format **stl** sous le nom **cubecreux.stl** prêt à être imprimé.

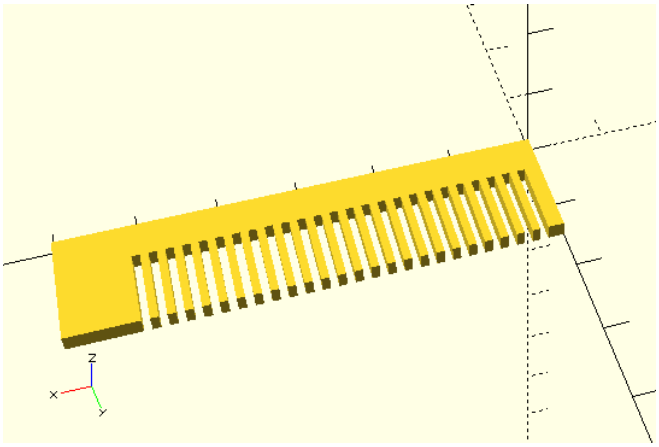
Astuce : utilisez 1 cube et 2 sphères : une pour évider le centre, l'antre pour lisser les côtés. Il sera nécessaire d'utiliser des compostions de fonctions.

4. activités

4.1 Un peigne

La création d'un objet se fait de manière générative en utilisant les structures de contrôle de l'algorithmique (séquence, répétition)

Votre premier objectif va être de composer « **un peigne** » (de longueur 60 et de largeur 15) en 3D comme ci-après en utilisant ces structures.

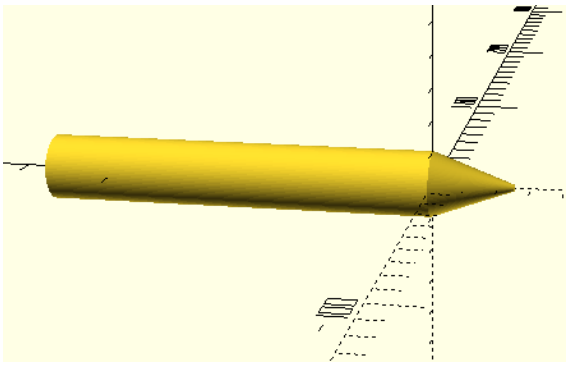


Nous allons définir un rectangle (*square*) sur lequel nous enlèverons un certain nombre de rectangles représentants les dents du peigne
Enfin, il faut utiliser une opération permettant de « donner de l'épaisseur » aux objets 2D selon l'axe Z (instruction *linear_extrude(h= …)*)

Une opération de répétition peut être utilisée !

4.2 Un crayon

Un moyen plus intéressant pour définir des formes en openSCAD est de définir des **modules**. Un module est l'équivalent d'une fonction dans les autres langages et accepte des paramètres en entrée.



Dans cette activité, vous allez définir un **crayon** sous forme de module composé d'un cylindre (de longueur 120 et de rayon 10 et d'un cône

Ce crayon peut être réutilisé autant de fois que nécessaire ..

Nota : améliorer votre module de manière à accepter 2 paramètres **h** et **r** permettant de définir la longueur et la largeur du crayon.

Enfin, il est possible de s'inspirer de fichiers STL déjà réalisés et les importer dans OpenSCAD (l'usage du plugin STL vers OpenSCAD (*http://www.thingiverse.com/thing:62666*) est recommandé).

5. adresses utiles

- **Syntaxe OpenSCAD en ligne** : http://www.openscad.org/datasheet
- **openSCAD UserManual:** https://en.wikibooks.org/wiki/OpenSCAD_User_Manual

- **Plugin Inkscape vers OpenScad :** http://www.thingiverse.com/thing:25036
- **Plugin STL vers OpenSCad** : http://www.thingiverse.com/thing:62666

6. solution des exercices

Premier exercice

```
$fn = 50;

intersection(){
    difference() {
        cube (3 , center=true) ;
        sphere (2 , center = true) ;
    };
    sphere(2.2, center = true);
}
```

Activité Peigne

```
// donner de l'épaisseur aux objets 2D
linear_extrude(2, center= false)

difference() {
    // dessiner un rectange
    square([60,15], center = false);
    // répeter 25 fois
    for (i = [1:25]) {
        // effectuer une translation
        translate([i*2,5,0]) {
            square([1,10]);
        }
    }
}
```

Activité Crayon

```
module crayon() {
    union(){
        cylinder(h=25, r1=10, r2=1, $fn=100);
        translate([0,0,-120]) {
            cylinder(h=120, r1=10, r2=10, $fn=100);
        }
    }
}


rotate([90,0,0]) {
    crayon();
}
```