Fabrication Numérique

Modélisation avec OpenScad

1

Objectifs

Comprendre les concepts de la modélisation type CAO

- Utilisation de la composition de primitives
- Produire des objets simples par programmation

Logiciel OpenScad

- Logiciel libre: http://www.openscad.org/
- Propose un langage de programmation intégré
- Fournit des maillages de qualités aptes à être utilisés en impression 3D ou découpe laser

Mais…

- N'est pas un logiciel pour réaliser des images de synthèse
  - Pas de prise en compte réelle du point de vue
  - Pas d'éclairage ni de matériaux complexes
- C'est de la programmation donc la syntaxe est stricte : L != l !!!

Documentation en ligne

Téléchargement d'OpenScad

- http://www.openscad.org/downloads.html

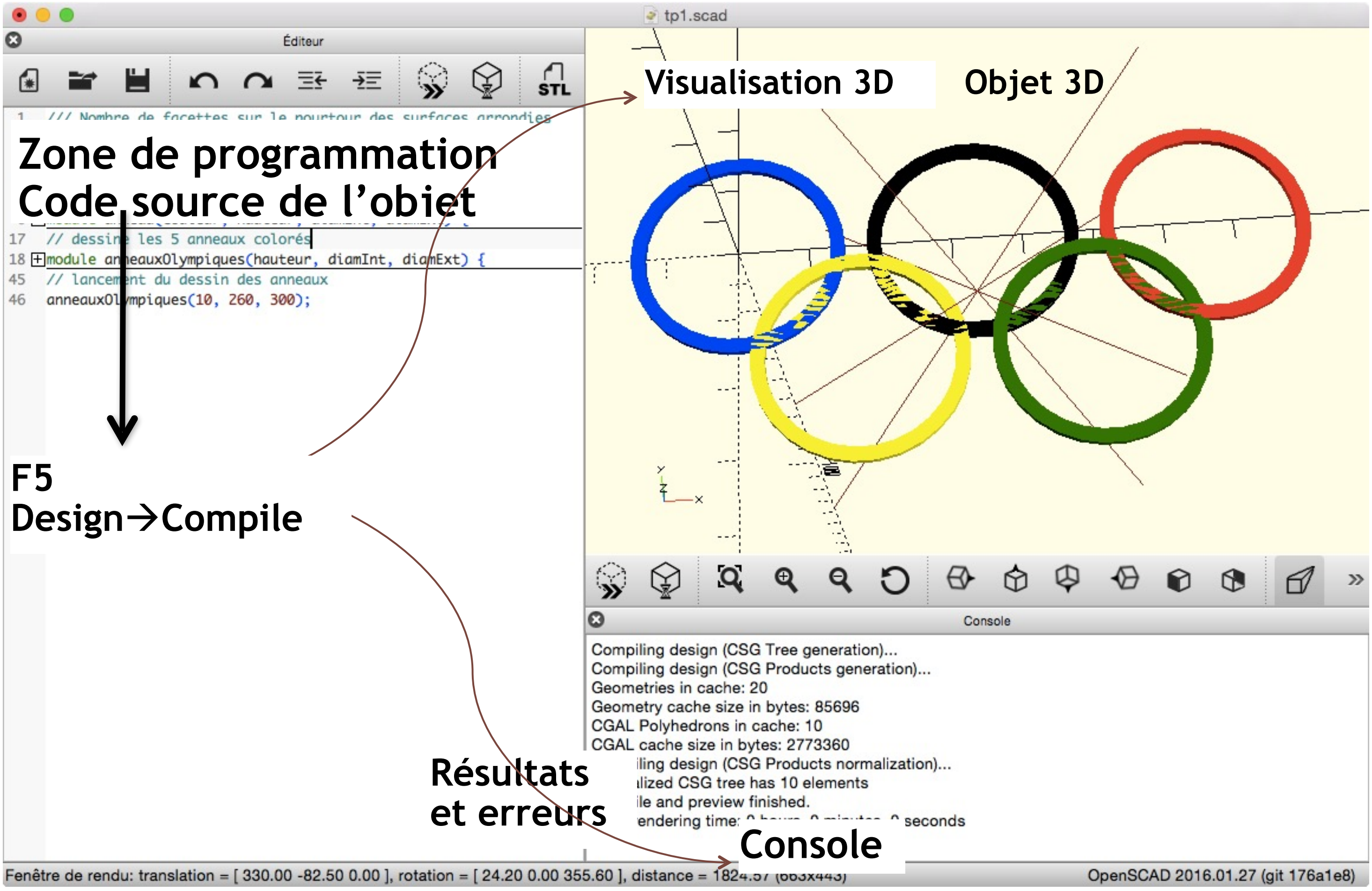
Tutoriels OpenScad

- https://en.wikibooks.org/wiki/OpenSCAD_User_Manual/First_Steps
- http://fablab-lannion.org/wp-content/tutos/openscad/#/intro

Documentation en ligne du langage

- https://en.wikibooks.org/wiki/OpenSCAD_User_Manual/The_OpenSCAD_Language

INTERFACE D'OPENSCAD



Fabrication Numérique : Openscad 4

Premiers pas

Fonctionnement

- On entre le « code » qui définit l'objet
  - Attention à la syntaxe stricte !!!!
- On compile ce code pour générer l'arbre CSG qui sera affiché
  - La compilation mais surtout la génération peut être longue
  - Les erreurs/informations s'affichent dans la console

Raccourci

- F5 permet de compiler

Exemples de codes: http://www.openscad.org/gallery.html

PREMIERS OBJETS

LES UNITÉS D'OPENSCAD SONT DES MILLIMÈTRES

PRIMITIVES DE BASE

- cube: trace des pavés
  - Syntaxe: cube([tx,ty,tz]) : cube([10,20,30])
  - Peut être centré sur 0,0,0 ou aligné sur les axes
    - Tester cube([10,20,30], center=true) vs cube([10,20,30], center=false)
- sphere:
  - Syntaxe: sphere(r=rayon) ou sphere(d=diamètre) : sphere(r=20)
  - La résolution peut être précisée ($fn,$fa,$fs) : sphere(r=20,$fn=100)
    - tester avec $fn=5 et $fn=1000 (affichage plus long non ?)
- cylindre:
  - Syntaxe: cylinder(r=rayon, h=hauteur) ou cylinder(r1=rayonbas, r2=rayonhaut, h=hauteur) →cones

Plus coloré

Application de couleurs

- color([r,g,b,a]) ou color ([r,g,b]) ou color (nom, a)
  - « r,g,b » définissent la couleur
  - « a » définit la transparence mais n'affiche pas les objets internes

Exemple

- ```
  color([0.2,0.6,0.6,0.5]) {
     sphere(r=2, $fn=100) ;
   }
  ```

La couleur n'est qu'indicative et n'est pas supportée par le format d'export (pour l'instant)
  ```

Placement des objets dans la scène

Transformations géométriques permettant de réaliser les opérations classiques:

- Translation :
  - translate ([tx, ty, tz]) { … }
- Rotation :
  - rotate ([angleX, angleY, angleZ]) { … }
- Mise à l'échelle
  - scale([echX, echY, echZ]) { … }

S'appliquent à ce qui est dans le bloc

- ```
  translate([10, 10, 10]) {
     sphere(r=2, $fn=100) ;
  }
  ```
  ```

Empilement des primitives et blocs

L'effet des transformations se cumule quand elles sont imbriquées

- Exemple : Translation, puis rotation, puis tracé du cube dans le repère local
  - Exemple :
```
translate([20, 20, 0]) {
  rotate([45, 45, 0]) {
    cube([10, 10, 10]);
  }
}
```

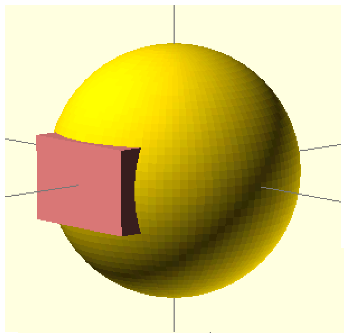
Blocs regroupent plusieurs opérations, exécutées en séquence mais au « même niveau »

- Définis par des { … } en respectant l'indentation
- Les instructions sont terminées par des ;

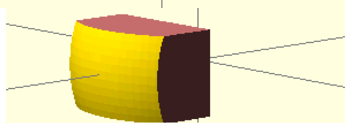
CSG : Constructive Solid Geometry

Opérations de composition booléenne

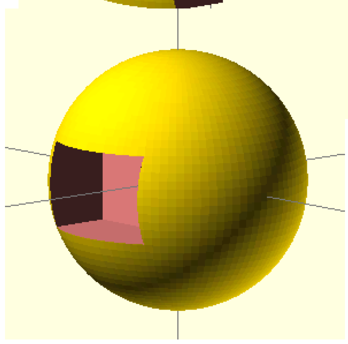
- Portent sur des éléments ou des séquences {…}
- Union :
  - ```
    union() {
      elements;
    }
    ```



- Intersection:
  - ```
    intersection() {
      elements;
    }
    ```



- Différence:
  - ```
    difference () {
      forme de départ;
      forme à soustraire;
    }
    ```

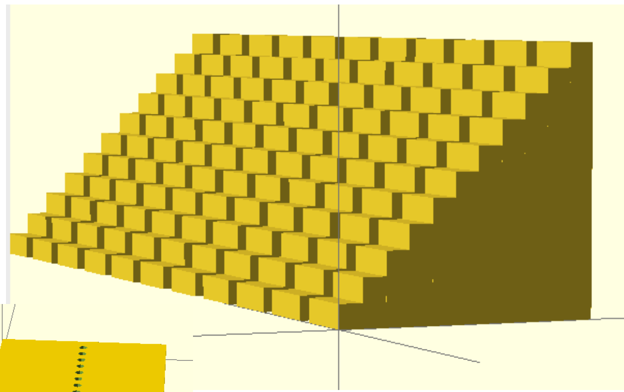

    ```

Exercice : object simples

Créer des objets simples

- Un cube
  - Déplacer le centre (center=false)
- Une sphère

Modifier leur couleur

- Jouer sur la transparence

Déplacer les objets pour obtenir un objet plus complexe

- Tester la non indépendance rotation/translation

Les composer

- Tester
- Avec union, intersection, difference, etc.

Les variables

Pas de définition des types, typage dynamique (fonction des affectations)

Les variables peuvent contenir :

- int
- float
- vecteurs
- chaînes de caractères
- etc. (voir doc)

Syntaxe:

- nom = valeur ;
- nom = fonction(...) ;

Exemple

- couleur = [0.4,0.3,0.9] ;
- Usage : color(couleur)   // par exemple

Boucles for

- Permet de répéter une séquence en faisant varier la valeur d'une variable
- Syntaxe générale:
  - ```
    for (variable = vecteur)  {
        partie qui sera répétée;
     }
    ```
  - le vecteur peut être :
    - Une liste d'éléments : [1, 3, 67] : 3 éléments
    - Un intervalle avec pas de 1 [min:max] : [-10:15] : valeurs -10, -9, ...14, 15 : 26 éléments
    - Un intervalle avec incrément != 1 [min:incr:max] : [-1:0.2:1] : valeurs -1, -0.8, -0.6 ... 0.8, 1 : 11 éléments
    ```

Boucles for

Exemples

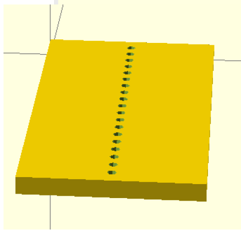
- ```
  for (x=[0, 20, 40])  {
     translate([x, 0, 0]) {
       cube([10, 10, 10]);
     }
  } // trois translations et 3 cubes
  ```
- ```
  for (x=[0:20])  {
     translate([x, x, 0]) {
       sphere(r = 1);
     }
  }  // 21 translations et 21 sphères
  ```

Exercice : escalier, plaque trouée



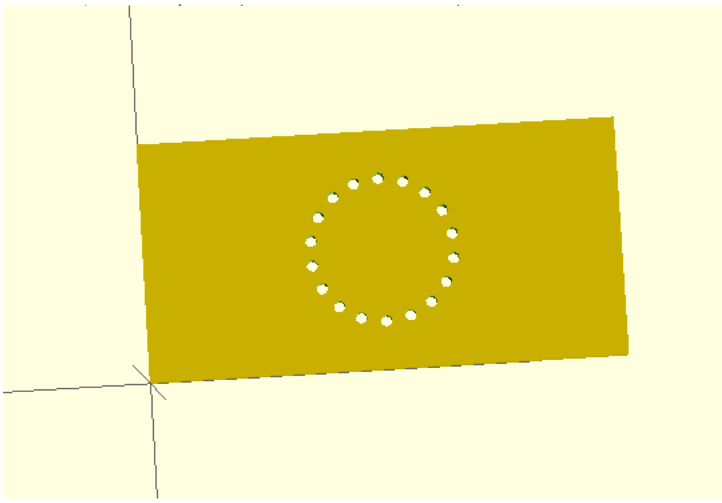
Réaliser un escalier

- ☐ Une seule volée de marches
- ☐ Plusieurs escaliers

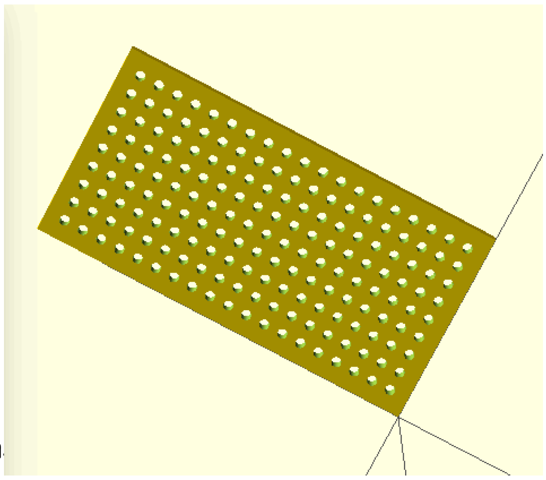


Créer une plaque trouée

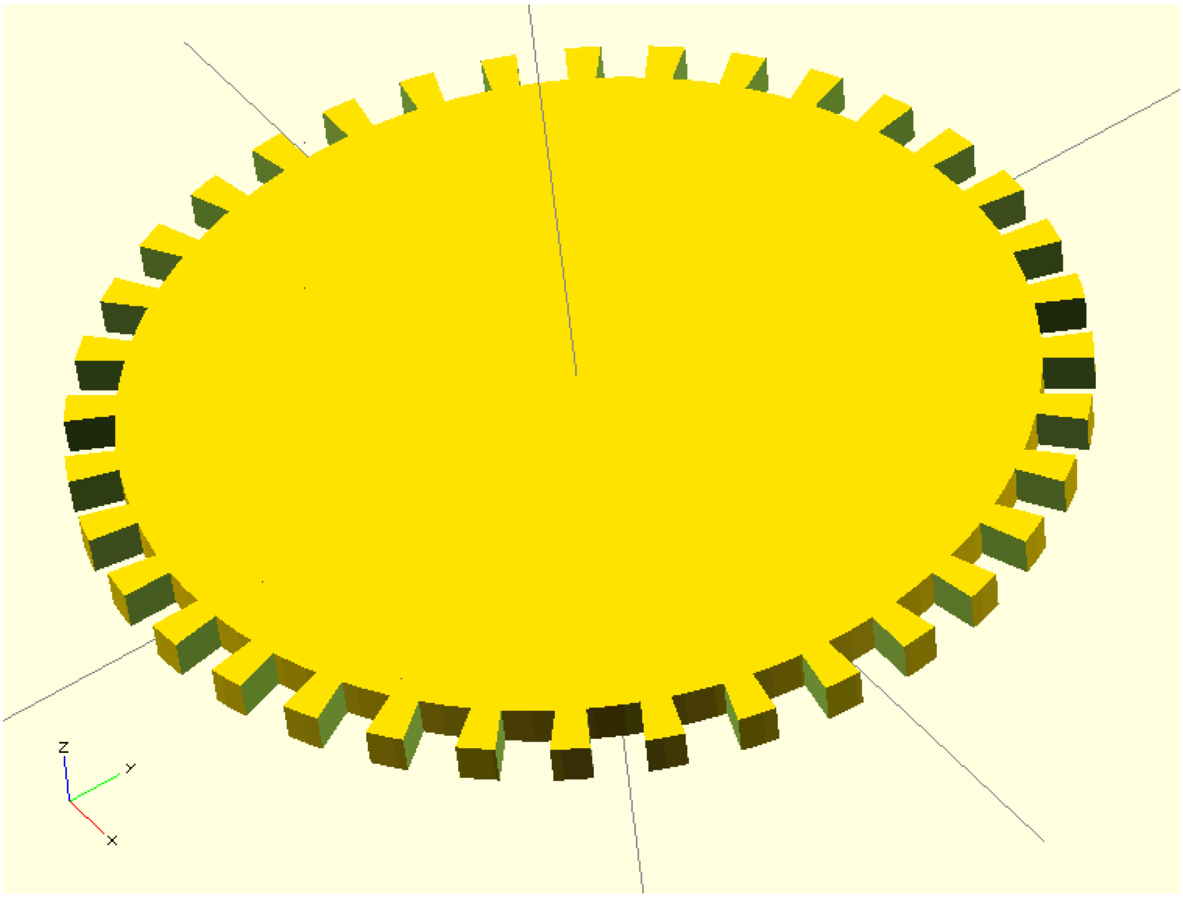
- ☐ Avec une seule ligne de trous
- ☐ Trouée sur toute sa surface



rique : Open



Exercice : Roue dentée



Fabrication Numérique : Openscad

16

Instructions conditionnelles

Vont permettre d'effectuer des actions en fonction de contenu de variables

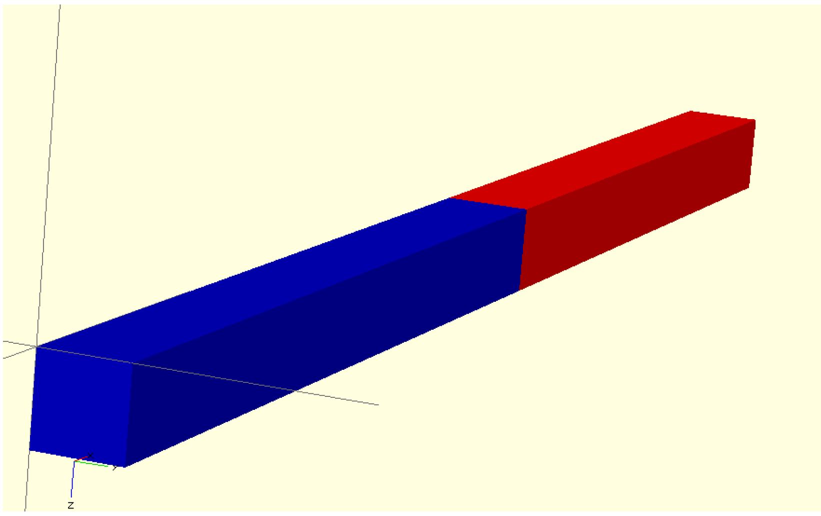
Syntaxe

```
if (condition) {
  ...bloc exécuté si la condition est vraie ...
} else {
  ...bloc exécuté si la condition est fausse ...
}
```

- Attention !!!!!!! test d'égalité s'écrit : ==

Exercice

Couleur conditionnelle



18

Création de modules (obligatoires)

Les modules permettent de créer des objets graphiques réutilisables

Création

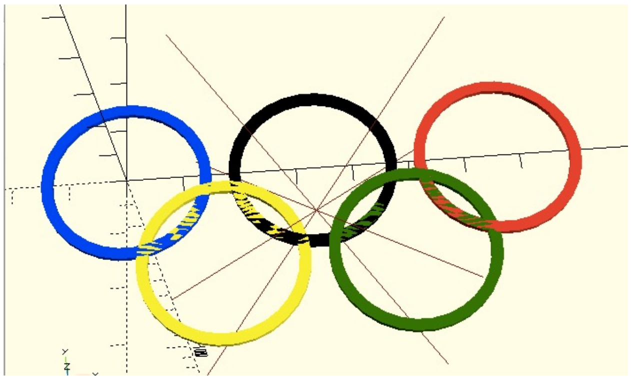
```
module engrenage(v, scl) { // paramètres v et scl seront typés par les arguments
   … contenu du module, v et scl sont des constantes dans le module …
}
```

Appel

engrenage([10,0,5], 3.4) → arguments v <- [10,0,5] et scl <- 3.4

Créer des modules à l'origine et de taille unitaire (permettra de gérer les mises à l'échelle)

EXERCICE : ANNEAUX OLYMPIQUES



MODULE anneau

- 3 paramètres
  - hauteur
  - diamètre intérieur : diamInt
  - diamètre extérieur : diamExt
- Primitive
  - cylinder(h = hauteur, d = diame, center = true)

    (center positionne la primitive par rapport à l'axe des Z, voir doc)

Exercice : Anneaux olympiques

Module anneauColore

- Ajout du paramètre couleur

5 anneaux

- Module anneauxOlympiques
- Appelle 5 fois le module anneauColore
- Export au format stl

Modélisation 2D

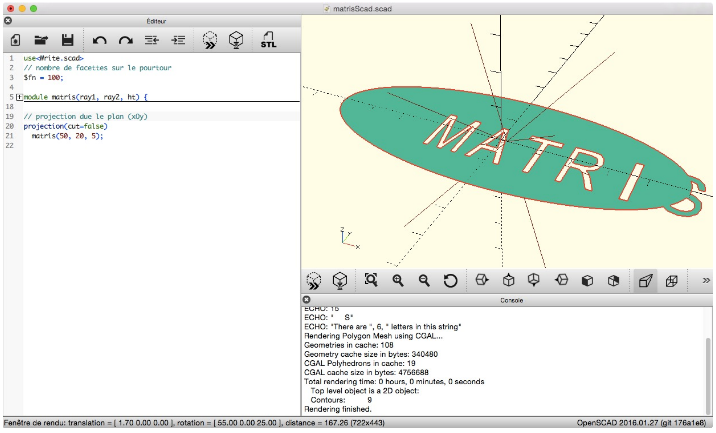
Plaque avec incrustation de caractères

- Utilisation d'une librairie « write.scad »
  - http://www.thingiverse.com/thing:16193
- Polices de caractères
  - Letters.dxf
- Opérations booléennes
- Projection selon l'axe des z
  -
    ```
    projection() {
        ... forme à projeter ...
    }
    ```

Modélisation 3D et 2D

Utilisation d'OpenScad

- Pour créer un modèle 3D
  - export au format stl
- Pour créer un modèle 2D
  - export au format dxl
- Pour positionner des primitives géométriques
  - translation, rotation, mise à l'échelle
- Pour combiner les primitives entre-elles
  - union, intersection, différence



Fabrication Numérique : Openscad

23