OpenSCAD
modèle de programmation

Voici le langage propre à OpenSCAD, c'est un langage de description de formes tridimensionelles simple et facile à appréhender. Trouvez ici un modèle de programme pour créer une pièce simple aux bouts arrondis et avec des perforations.

Nous trouvons ici les suivantes fonctions essentielles :

- **cube :** qui permet de réaliser des cubes ou des prismes avec les coordonnées X, Y, et Z.

- **cylinder :** qui comme son nom l'indique produit des cylindres utilisant les variables "h" pour hauteur et "r" pour rayon ou "d" pour diamètre.

- **translate :** pour positionner un objet selon ses coordonnées X, Y, et Z.

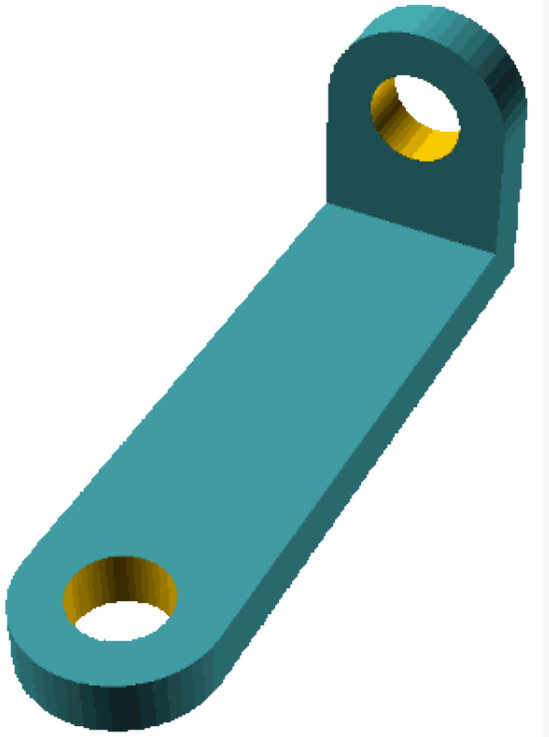
- **rotate :** fonction qui permet de de définir l'inclinaison d'un objet dans l'espace indiquant des valeurs d'inclinaison pour les coordonnées X, Y et Z.

- **différence :** qui crée une "différence d'état, permettent ainsi à un objet qui traverse un autre de créer un perforation.

  Nous avons ici, à fin de rendre le programme plus compréhensible, crée des **"modules"** qui sont en quelque sorte des sous programmes contenant chaque un les éléments qui composent l'objet.

  L’utilisation des variables prédéfinies en début du programme permet de changer les dimension de la pièce tout en maintenant sa forme.

```
// initialisation des variables
x1 = 10;
y1 = 30;
z1 = 3;

x2 = 10;
z2 = 10;

// nombre de facettes des circonférences

$fn=50;

// définir la circonférence des
terminaisons
de la pièce
module ending(){
cylinder(d=x1,h=z1);
}

//définir les 2 perforations
module perf(){
cylinder(d=x1/2,h=z1);
}

// assemblage de la pièce
barres et arrondis
module hook(){
cube([x1,y1,z1]);
translate ([x1/2,00,00]) ending();

translate ([00,y1,00]) cube([x2,z1,z2]);

translate ([x1/2,y1+z1,z2])
rotate ([90,00,00]) // angle arrondi
ending();
}

module perf_hook(){
difference() {
hook();
translate ([x1/2,00,00]) perf();
translate ([x1/2,y1+z1,z2])
rotate ([90,00,00]) perf(); //angle perf

}}

perf_hook();
```



Langage OpenSCAD

Syntax

var = value; module name(…) { … }
name(); function name(…) = …
name(); include <….scad> use <….scad>

2D

ircle(radius | d=diameter)
square(size,center)
square([width,height],center)
polygon([points])
polygon([points],[paths])
text(text, size, font,
halign, valign, spacing,
direcction, language,
script)

3D

sphere(radius | d=diameter)
cube(size, center)
cube([width,depth,height],
center) cylinder(h,r|d,center)
cylinder(h,r1|d1,r2|d2,center)
polyhedron(points, triangles,
convexity)

Transformations

translate([x,y,z])
rotate([x,y,z])
scale([x,y,z])
resize([x,y,z],auto)
mirror([x,y,z])
multmatrix(m)
color("colorname",alpha)
color([r,g,b,a])
offset(r|delta,chamfer)
hull()
minkowski()

Boolean operations

union()

difference()

intersection()

Modifier Characters

* disable

! show only

highlight / debug

% transparent / background

// and /* ,,, */ comment

Mathematical

abs sign sin

cos tan acos

asin atan atan2

floor round ceil

ln len let log

pow sqrt exp

rands min max

Functions

concat
lookup
str
chr
search
version
version_
num
norm
cross
parent_
module(idx)

Other

echo(…) for (i = [start:end]) { … }

for (i = [start:step:end]) { … }

for (i = […,…,…]) { … }

intersection_for(i = [start:end]) { … }

intersection_for(i = [start:step:end]) { … }

intersection_for(i = […,…,…]) { … } if (…) { … }

~~assign (…) { … }~~

import("….stl")
linear_extrude(height,center,convexity,twist,slices,scale)

rotate_extrude(angle,convexity)

surface(file = "….dat",center,convexity)

projection(cut)

render(convexity)

children([idx])

Special variables

$fa minimum angle

$fs minimum size

$fn number of fragments

$t animation step

$vpr viewport rotation angles in degrees

$vpt viewport translation

$vpd viewport camera distance

$children number of module children

List Comprehensions

Generate [for (i = *range*|*list*) i]

Conditions [for (i = …)
if (condition(i)) i]

Assignments [for (i = …)
let (assignments) a]