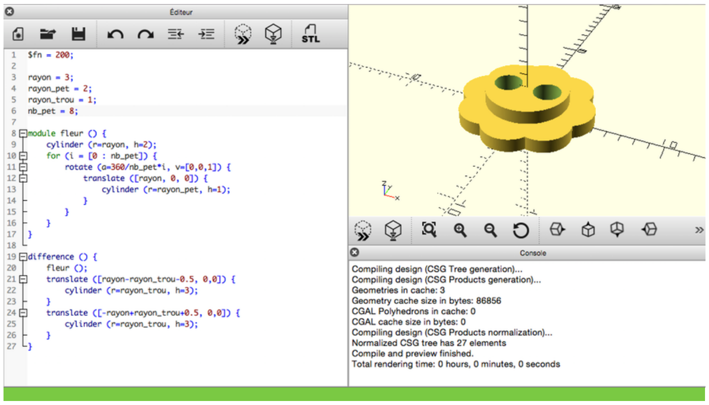
From code to 3D printing



```
$fn = 200;

rayon = 3;
rayon_pet = 2;
rayon_trou = 1;
nb_pet = 8;

module fleur () {
    cylinder (r=rayon, h=2);
    for (i = [0 : nb_pet]) {
        rotate (a=360/nb_pet*i, v=[0,0,1]) {
            translate ([rayon, 0, 0]) {
                cylinder (r=rayon_pet, h=1);
            }
        }
    }
}

difference () {
    fleur ();
    translate ([rayon-rayon_trou-0.5, 0,0]) {
        cylinder (r=rayon_trou, h=3);
    }
    translate ([-rayon+rayon_trou+0.5, 0,0]) {
        cylinder (r=rayon_trou, h=3);
    }
}
```

```
Compiling design (CSG Tree generation)...
Compiling design (CSG Products generation)...
Geometries in cache: 3
Geometry cache size in bytes: 86856
CGAL Polyhedrons in cache: 0
CGAL cache size in bytes: 0
Compiling design (CSG Products normalization)...
Normalized CSG tree has 27 elements
Compile and preview finished.
Total rendering time: 0 hours, 0 minutes, 0 seconds
```

🕘 2h30 | 👥 10 | ●●●●○

Public

7-12 y-o

12-15 y-o

adult

Software

OpenSCAD

Description

Learning goals:

- **Understand the basics of programming**

The goal of this workshop is to use 3D printing to better understand the basics of code :

- variables
- loops
- modules
- functions

Timing

1. Introduction and presentation of 3D printing - 15 mins

Introduction of the world of 3D printing with video (http://www.dragontape.com/#!/6319966877908992) and presentation.

2. Overview of the activity - 5 mins

Briefly explain the flow of steps for the day's activity.
What will the participants do? : Create a clothing button which is shaped like a flower.
Which software will be used? : OpenSCAD

3. Launch OpenSCAD on everybody's computer - 2 min

4. Introduction to OpenSCAD - 20 min

Provide some explanations of the programming syntax - with cheatsheet opened (http://www.openscad.org/cheatsheet/)

- cube(5); cube([3,1,2]); cube(5, center);
- translation / rotation
- other types of primitive solids (cylinders, spheres, etc.)
  take note of $fn (specifying number of faces)
- boolean operations (addition, subtraction, intersection)
- variables

5. The flower - 25 min

- cylinder in the centre
- cylinders around the centre to make the petals (modules)

- cylinders for the holes

To go further or get more advanced, invite participants to:

- variables for the size of the petals
- variables for the size of holes and their spacing
- multiplication with for loop
- variable for the number of petals
- try to make a stable model regardless of changing values

6. Example of code

Code to make a button flower :

```
fn = 200;

radius = 3;
radius_pet = 2;
radius_hole = 1;
nb_pet = 8;

module flower () {
  cylinder (r=radius, h=2);
  for (i = [0 : nb_pet]) {
    rotate (a=360/nb_pet*i, v=[0,0,1]) {
      translate ([radius, 0, 0]) {
        cylinder (r=radius_pet, h=1);
      }
    }
  }
}

difference () {
  flower ();
    translate ([radius-radius_hole-0.5, 0,0]) {
        cylinder (r=radius_hole, h=3);
    }
    translate ([-radius+radius_hole+0.5, 0,0]) {
      cylinder (r=radius_hole, h=3);
    }
}
```

7. Finishing the 3D model - 10 min

Participants play with the variables (including $fn). Once they are satisfied, they can export the STL file

8. Import the files in Tinkerine or Cura - 5 min

Open the preferred slicer software
Import the STL file
Adjust the print settings (fill, wall thickness, speed, height, ...)
Save the file (.Gcode) to the SD card

9. 3D Printing - 30–45 mins

Insert the SD card in the 3D printer
Select the previously saved file (.Gcode)
Start the print!

10. Feedback - 5 min

Ask the participants to provide feedback on the workshop while the models are being printed.

Partcipant's guide

1. Code to make a button flower :

```
fn = 200;

radius = 3;
radius_pet = 2;
radius_hole = 1;
nb_pet = 8;

module flower () {
  cylinder (r=radius, h=2);
  for (i = [0 : nb_pet]) {
    rotate (a=360/nb_pet*i, v=[0,0,1]) {
      translate ([radius, 0, 0]) {
        cylinder (r=radius_pet, h=1);
      }
    }
  }
}

difference () {
  flower ();
    translate ([radius-radius_hole-0.5, 0,0]) {
        cylinder (r=radius_hole, h=3);
    }
    translate ([-radius+radius_hole+0.5, 0,0]) {
      cylinder (r=radius_hole, h=3);
    }
}
```