Initiation to 3D Printing – Practical exercises 2

Sylvain Lefebvre and Camille Schreck, ENSEM 2020-2021

Tuesday 24th January, 2023

1 Important information

- I recommand to write the code in C++ (the templates and corrections will be in C++). But you can also use C, Python, or JAVA.
- At the end of the session, send the **code and GCode of exercises 3, 4, 5, 6 and 7 (bonus)**. The files should be in a single folder called **TP1_[nom][prenom]** and compressed into a ZIP (or tar.gz..) file to:
  - sylvain.lefebvre@inria.fr
  - camille.schreck@inria.fr

  with the mail subject **ENSEM: TP 1 [nom][prenom]**

2 Useful Links

- To write and test GCode `https://icesl.loria.fr/webprinter/` (older version: `http://shapeforge.loria.fr/vrprinter`)
- Another GCode viewer `http://gcode.ws`
- List of GCode instructions `http://marlinfw.org/meta/gcode/`

3 Exercise: filling the square (zigzag infill)

1. Consider a square with dimensions 40×40mm. Implement the filling of the square (slice) with a zigzag infill. Make sure that the zigzag fills (as close as possible) 100% of the inside of the square Beware of overlaps!

2. Modify the program to output 50 layers of thickness 0.2mm, for an object of 10mm height in total. The zigzag main direction rotated 90 degrees at each slice (left/right, then front/back).

4 Exercise: filling the square (contour parallel infill)

1. Consider again a square with dimensions 40×40mm. Implement the filling of the square (slice) with a contour parallel infill.

2. Modify the program to output 50 layers of thickness 0.2mm, for an object of 10mm height in total.

5 Exercise: mixing zig zag and contour parallel

1. Consider again square with dimensions 40×40mm. Implement the filling of the square with two parallel contours, followed by the zig-zag infill.
2. Modify the program to output 50 layers of thickness 0.2mm, for an object of 10mm height in total.

6 Exercise: filling the hemisphere

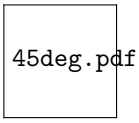
1. Implement a program outputting the GCode of of circle of 20mm radius on one layer. It will be filled with two contour parallel tracks, followed by a zigzag infill.
2. Bonus: Implement a program outputting the GCode of an hemisphere of 20mm radius with 0.2mm layers. Each layer will be filled with two contour parallel tracks, followed by a zigzag infill.

7 Sparse filling a cube

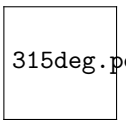
7.1 Three sets of parallel lines in a square

We are going to fill a square of 36×36 mm with three sets of parallel lines:

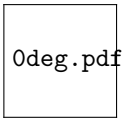
1. The first set is at a 45 degree angle with a spacing of 4mm.



2. The second set is at a -45 degrees angle with a spacing of 4mm.



3. The last set is at an angle of 0 degree angle with a spacing of 4mm.



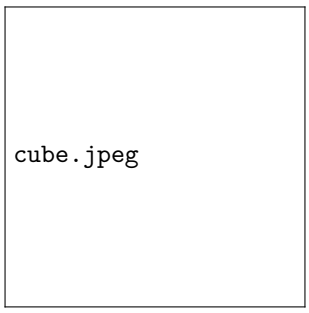
Make sure the 4mm spacing can be adjusted from a variable.

7.2 Progressive offset in the cube

We are going to fill a cube of dimensions $36 \times 36 \times 36$ mm

Progressively offset the lines at each layer, moving them sideways (to their right) by half a nozzle (typically 0.2mm). Generate GCode. You should now see three sets of angled walls forming closed 3D cells. For an illustration and more information refer to this URL:

`http://sylefeb.blogspot.com/2015/07/3dprint-3d-infilling-faster-stronger.html`



8 Miscellaneous: sample code C++

```cpp
#include <iostream>
#include <fstream>
#include <cmath>  // use constant M_PI to get the value of pi

int main () {
    std::ofstream file;
    file.open ("square.gcode");
    // header
    file << "G21" << std::endl;  // dimensions in milimeters
    file << "G90" << std::endl;  // absolute positioning
    file << "G28" << std::endl;  // homing

    // exercise code
    file.close();
    return 0;
}
```

In Linux, compile the above program (contained in a file main.cpp) with:

```
g++ main.cpp -o main
```