

Gel Printer Software HowTo

Jäkel, Anna Christina

April 2020

Contents

1	Brief Overview	2
2	Software (Preparations)	2
2.1	How to Install Python	2
2.2	Using the Command Line Interface	3
2.3	Getting the Code	4
3	G-code Generator Code	4
3.1	Steps to generate G-Code	4
3.2	File Description	5
3.3	The SVG File	5
3.4	Description of all Input Parameters	7
3.5	Execute Python Code	10
4	Working with the G-code	10

1 Brief Overview

The GitLab repository *Bioprinter* hosted on GitHub (<https://github.com/julia-mueller/bioprinter>) contains the complete Python 3.0 source code written by Julia Müller and Anna Jäkel to produce G-Code to drive a hydrogel loaded 3D printer.

Our Python code enables the conversion of an svg-file containing the desired hydrogel structure into machine readable G-Code. It is further possible to specify a variety of printing parameters, e.g. nozzle diameter, feedrate, or volume that should be extruded. In Figure 1 an exemplary svg-file and the 3D object produced with this svg-file are shown.

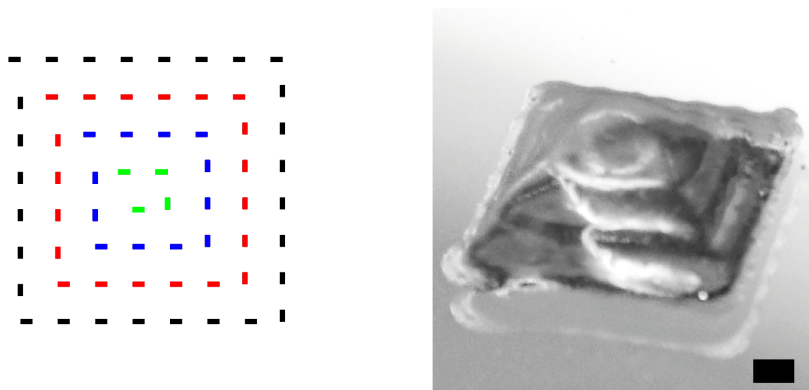


Figure 1: The input svg-file where each short line represents a single printed voxel produces a 3 dimensional printed pyramid structure (scalebar 1mm). Multiple layers are indicated by colors.

2 Software (Preparations)

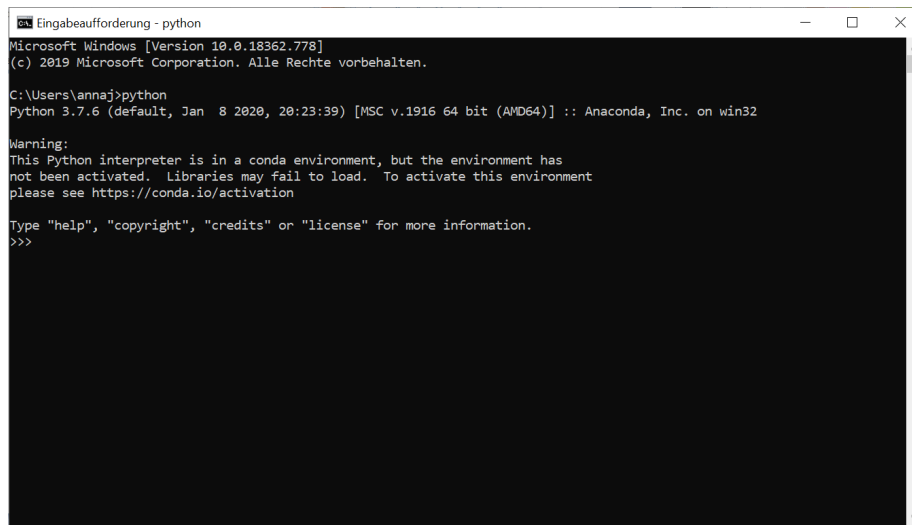
If you are a complete beginner to Python programming, the first sections will guide you through the installation process. If you are familiar working with Python you can directly continue with Section 2.3.

2.1 How to Install Python

Our code is written in Python 3.0. You can choose between installing Python individually, or in combination with a package management system (Anaconda). In both cases it is very important that you use Python 3 and not Python 2, because there are some syntax differences and our code would not compile without errors. We decided for an implementation in Python 3 since Python 2 is in EOL (End of Life) status and further use is not recommended.

If you choose to install it manually, you first have to download Python from the

official Python website <https://www.python.org/downloads/>. After downloading the Python 3 installation files you have to follow the installation instructions of the installer which will open after a double click on the downloaded file. After your installation finished successfully you can type `python` in a command line interface. You can open this on windows when you type `cmd` in the search line or after you used the key combination "windows + R". This should look like Figure 2. As a response the version of python should be shown. This works similar for macOS, here the command line interface can be found when searching for `terminal`. You can run python code by typing "`python mycode.py`",



```
Eingabeaufforderung - python
Microsoft Windows [Version 10.0.18362.778]
(c) 2019 Microsoft Corporation. Alle Rechte vorbehalten.

C:\Users\annaj>python
Python 3.7.6 (default, Jan 8 2020, 20:23:39) [MSC v.1916 64 bit (AMD64)] :: Anaconda, Inc. on win32

Warning:
This Python interpreter is in a conda environment, but the environment has
not been activated. Libraries may fail to load. To activate this environment
please see https://conda.io/activation

Type "help", "copyright", "credits" or "license" for more information.
>>>
```

Figure 2: Command line on windows

where *mycode.py* is your file that contains python code. Python code can be written in any text editor, e.g. Atom. You have to save the file with the ending *.py*.

However, we recommend to download the package management system Anaconda with Python 3 instead of the individual installation of Python 3.0. Anaconda comes with additional packages and facilitates the download of further packages on the fly. Otherwise you would have to manually download and install those. With Anaconda all the packages required for our code are automatically installed. You can download Anaconda here: <https://www.anaconda.com/products/individual>. Also take care to install Anaconda with Python 3.0.

2.2 Using the Command Line Interface

The command line is a powerful tool when using your computer. It is a text interface, which takes commands and passes them to the operating system of

your computer. There are a lot of useful commands but in the following only the most important ones for our usecase are presented:

- `cd path` : changes the directory to the indicated path
- `cd ..` : moves you one directory stream upwards
- `dir` : shows the contents of your folder (use `ls` instead for linux)
- `python mycode.py` : executes a python script

2.3 Getting the Code

The source code for the gel printer is available on:

<https://github.com/julia-mueller/bioprinter>.

You can either download all files and folders manually (see Section 3.2) of this Github directory to a directory on your local computer or you use git to do that for you. For a short introduction to git see our *Instruction for GitLab* PDF (also available in our github project).

3 G-code Generator Code

After a brief step-by-step guide this section explains the python files in our directory. Further it shows the structure of the input svg-file, and possible input parameters in detail. At last support on the execution of our code is given.

3.1 Steps to generate G-Code

To execute the code to generate your G-Code you have to do the following steps:

1. **Prepare SVG-File:** Draw your desired structure and save it as svg. Save your svg-file with the print structure in the folder *svgfiles*. You have to make sure that the svg file is in a specific format. See Section 3.3 on how to design your structure.
2. **Prepare Parameterfile:** Include the name of your new svg-file in your parameterfile. Save your custom parameterfile in the folder *parameterfiles*. We explain all parameters that can be adjusted in Section 3.4.
3. **Execute generator code:** Use the command line to call the python script with `python generator.py parameterfiles/'my_file.yaml'` directly together with your parameterfile. How this is done will be explained in Section 3.5.
4. **Get G-Code:** After successful conversion find your G-Code in the folder *gcodefiles* and copy it on a stick or send it to your printer. Additional information about this is given in Section 4.

3.2 File Description

To successfully execute the Python code for converting svg-files to G-code make sure the following files are all saved in one location:

- `executor.py`: this file starts the G-Code generator
- `generator.py`: input and output file handling
- `routinегenerator.py`: translation of svg items to G-Code
- `constants.py`: repetitive printer movement commands

Also the following folders are necessary:

- `parameterfiles`: contains your input parameterfile
- `gcodefiles`: will contain your output G-Code files
- `svgfiles`: contains your input svg-file

3.3 The SVG File

We recommend to use a vector graphic program like Inkscape, Adobe Illustrator or similar to design the print layout. Again we first provide a short description followed by detailed explanations for all steps.

Summary SVG-File

In short your svg-files should follow these rules:

- open px-defined new artboard
- draw lines with stroke width $\neq 1$
- line length similar to nozzle diameter for spheric voxels
- each layer is indicated by one color
- white lines will be ignored
- ungroup everything
- the lines may not have an id / name
- save with enough decimal places

Drawing Area

All vector graphic programs we used showed their origin in (x|y) in the upper left corner. In contrast, the origin of the print area of our 3D printer is in the lower left corner. So you should bear in mind to plan your print accordingly, to avoid a horizontally mirrored result. On the printbed we included an offset in x and y direction of 30mm to gain a larger home area and the option of larger sample holders. Hence all points are printed at their respective location of the svg file (i.e. distance to the upper left corner) plus (30mm|30mm) from the lower left corner on the printbed.

Point-wise Representation

Structures are printed voxel-based (pointwise). Each voxel is represented in the svg-file as a short line. To gain sphere like voxels we decided to use line lengths comparable to the nozzle diameter (0.238 mm worked well for 0.2mm nozzles). Dependent on your program it might scale your drawing area upon saving. To avoid unintended scaling, we recommend to open your drawing area in point or pixels. In the G-code 1 point/pixel corresponds to exactly 1 mm.

Note: It is important to use a different stroke width than 1, as otherwise this parameter will not be listed in the svg representation and the script cannot be executed. However for most of our print designs the svg model needs stroke widths around 0.2 to avoid an overlay of lines.

Note: So far it is only possible to read lines for our program, but we are working on the translation of polylines to G-code.

Layer Definition

Layers are indicated by the stroke color of a line. For each layer a hexcode is handed over to the G-code Generator via the parameter *colorlist*. All lines of this color will then be printed at this layer. With the usage of different colors it is possible to define different structures for each layer.

Note: Avoid the color white "FFFFFF" as this is used to exclude a layer:

The *colorlist* parameter defines which color should be printed in each layer. With our script it is also possible to assign different colors to different extruders. If you want to pause one extruder in a specific layer then you insert "FFFFFF" as color for this layer. A complete explanation on the input parameters can be found in Section 3.4.

File Save Options

Save the point- or pixelwise defined svg-file with enough decimal places (~ 5) to avoid loss of precision. Figure 3 shows the settings to save a file correctly in Adobe Illustrator 2020.

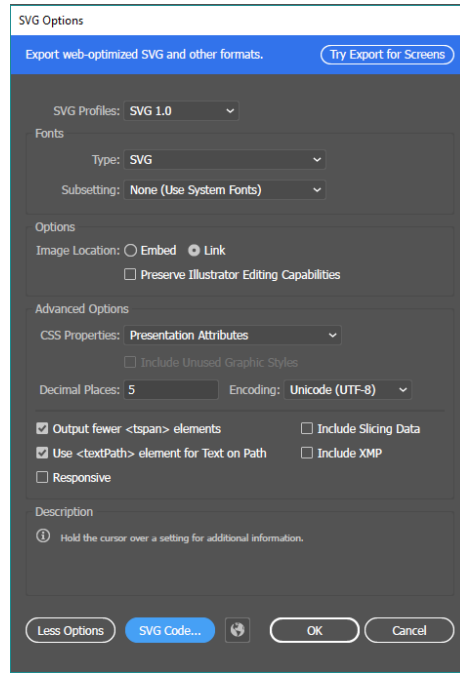


Figure 3: Options for saving a svg-file in Adobe Illustrator.

3.4 Description of all Input Parameters

You will find an exemplary parameterfile provided in the folder parameterfiles as mentioned before. Figure 4 shows how this looks like. All parameters that can be handed over to the Python script are listed and explained in the tables of this section:

- **Table 1:** Parameters that have to be adjusted for each print
- **Table 2:** Parameters that allow to fine tune the print results
- **Table 3:** Parameters for the initial setup of hydrogel printing with our code on your specific 3D printer
- **Table 4:** Global parameters, only for initialization

Table 1: Update these material parameters for each print (exemplary parameters)

parameter	explanation
inputfile	'name_of_svg_file.svg'
colourlist	[[000000,FF00FF,000000][000000,FFFFFF,00FF22]] [['color_extruder1_layer1', 'color_extruder1_layer1', ...] ['color_extruder2_layer1', 'color_extruder2_layer2', ...]] colors in Hexcode without # (e.g. 000000, FF0000) colour = layer → number of colors = number of layers all lines with this color are printed in this layer
nozzle.temp	45 nozzle temperature in °C, for our bioink we used 45°C
bed.temp	0 bed temperature in °C, if no heating required, set to 0
height	[1, 11] [thickness_object_slide, max_height_object_slide_holder]
red_delta_z	0.8 multiply calculated necessary z step per layer (reduction dependent on contact angle)
increase_volume_factor	1.0 multiply calculated necessary volume by this factor

Table 2: Update these material parameters to optimize your print results

parameter	explanation
scale_z	True or False determines if the nozzle-sample distance should be changed additional to red_delta_z by a linear factor
max_normal_layer	10 number of layers after which additional z height reduction is necessary, works only if scale_z = True
dim_z_later	0.9 similar to red_delta_z but considered only for layers after max_normal_layer is reached
scale_z_squared	True or False scale the z height by red_z_square · number(layers) ²
red_z_square	0.01 factor for the squared reduction
corner_adjustment	True or False corner adjustment due to belt slippage might be necessary, can be switched on

Table 3: Update these parameters for your own printer at first usage

parameter	explanation
syringe_radius	2.5 radius of your syringes in mm
nozzle_diam	0.1 nozzle diameter in mm
flowspeed	500 (worked best for our bioink. The flowspeed is translated to the G-code parameter F = feedrate and sets the print speed)
conversion_ratio_syringe	1.0 necessary only if you use different syringes for the external pump and your sample
offset	50.0 offset between the extruder in mm
homeheight	9.7 height in mm of the mechanic or electric endswitch that sets $z = 0$
slippage	0.5 value for the slippage of the printer in mm
startdis	30 offset from home position (0 0) before print starts in x and y to allow the installation of custom sample holders

Table 4: The parameterfile initializes the following global variables, these do not need to be changed manually

parameter	explanation
volume_gel	[0.0, 0.0, 0.0]
begin_of_print	0
z_safe_dis	3 height added to z during movements in mm

```

1 # -*- coding: utf-8 -*-
2 # 05.03.2019 Julia Müller
3 # revised by Anna Jäkel
4
5 # This .yaml file acts as a bibliography for all parameters that are needed.
6
7 #-----
8 # Filename of the inputfile:
9 inputfile : "pyramid_print_test.svg"
10
11 # List of strings with colour hexcodes (50 layer)
12 # for each colour print elements in that colour --> layers
13 # black, black, red, red, blue, blue, green, green
14 colourlist : ["000000", "000000", "000000", "FF0000", "FF0000", "FF0000", "0000FF",
15              "0000FF", "00FF00", "00FF00"]
16
17 # Name of the used liquid:
18 liquid : "Agarose F500"
19
20 # liquidparameters [flowspeed, nozzle temperature, bed temperature]:
21 liquidparameters : [500, 45, 0]
22
23 #volume [ , , ]
24 volume_gel : [0.0, 0.0, 0.0]
25 #-----
26
27 # Parameters for the printer:
28 #-----
29 # Radius of the syring:
30 syringe_radius : 4.5 # in mm
31
32 # Innerdiameter of the nozzle:
33 # possible values: 0.11(Gelb) | 0.15(Lavendel) | 0.20(Klar) | 0.25(Rot) | 0.33(Orange) | 0.41(Blau)
34 nozzle_diam : 0.20 # in mm
35
36 # Height of object slide and object slide holder
37 # [sheight=height of object slide, hheight=height of the object slide holder]
38 height : [1.1, 5.0]
39
40 # homehoefahrt:

```

Figure 4: Parameterfile

3.5 Execute Python Code

Open a command line interface and change to the folder where the python scripts and the above described folders are in. Then type in `python executor.py parameterfiles/'your parameter file name'.yaml`. In Figure 5 an example execution is shown.

When everything worked fine the message *All done!* will be shown, also the name of the outputfile is given and the volume will be shown that is needed for the print. If errors occur, please read the error code and compare it to the Troubleshooting document in the Github directory. A list of frequently occurring errors and recommendations are given there. Also refer to Section 3.3 to make sure your svg-file is in the correct format. If input parameters are missing, this will be noted in the error message and you have to update the parameterfile.

4 Working with the G-code

After you compiled the Python code and successfully created a G-Code file, you will find this G-Code file in the folder *gcodefiles* in a subfolder named after the creation date.

TaDaa! You now can copy this G-code to a SD card or USB drive and print it with a gel laden 3D printer.

Hardware: Construction files for our 3D printer setup are available on request. Protocols for the preparations of bioink and the general print setup can be found in the paper mentioned in the README of our Github repository.

```

MINGW64://nas.ads.mwn.de/ge29huq/Studentische Hilfskraftstelle E14 Simmel/g...
annaj@DESKTOP-BB75F8T MINGW64 //nas.ads.mwn.de/ge29huq/Studentische Hilfskraftst
elle E14 Simmel/gel-printer (master)
$ python executor.py parameterfiles/2020-02-20_Pyramid.yaml
--> All done!

Outputfile: w_2.0V_pyramid_print_step-on_squ-off_cor-on.gcode

Volume is: [0.0, 5.92, 0.0]

annaj@DESKTOP-BB75F8T MINGW64 //nas.ads.mwn.de/ge29huq/Studentische Hilfskraftst
elle E14 Simmel/gel-printer (master)
$

```

Figure 5: Compilation of a parameterfile.

Customization Options: The name of the G-code file is by default *nozzlediameter-svg-filename.date*. If you like to change how the filename is built, this can be done in the Python file `generator.py` in the function `get_filename`. We recommend to include at least the svg filename and the nozzle to your filename to facilitate your print.

We would be happy to publish any contributions to our code for other users. If you like to do so, just send us a merge request with your code additions/changes.

```

1 ;FLAVOR:RepRap
2 ;TIME: 14:09:02
3 ;Filament used: Agarose F500;Layer height: 0.2
4 ;Nozzle used: 0.2
5 ;Generated by Julia's Generator
6 ;with the following parameterfile: parameterfiles/2020-02-20_Pyramid_test.yaml
7
8 M107 ; turn fan off
9 M302 ;allow cold extrudes
10 G21 ;metric values
11 G90 ;absolute positioning
12 M83 ;set extruder to relative mode
13 M92 E3200; 1 mm syringe pump way is reached after 3200 motor steps
14 M117 Home all axes ;Put home message on LCD screen
15 G98 ; use absolute positioning like always
16 G28 X0 ;move X to min endstop
17 G28 Y0 ;move Y to min endstop
18 G28 Z0 ;move Z to min endstop, get nozzle to 0 now
19 G1 Z5G1 X10 Y10 ;move away from spring
20 G92 Z14.7 ;set homebed to zero
21 ;M117 Printing cool gelly stuff ;Put printing message on LCD screen
22 G1 X10 Y10 E0.2; Extrude to make sure gel is in the nozzle
23 G1 Z5.0; move z-stage down to not crash to the sample holder
24
25 G1 F5000 X40 Y40
26 M117 Layer 1
27
28 ;change to first extruder at the beginning of each layer
29 T1
30
31 ;GCODE of lines
32 ;M117 Print lines...
33
34 G1 F3000 X31.155 Y32.088997 E0
35 G1 Z1.1 E0.0
36 G1 F500 X31.393 Y32.088997 E0.00035259259259259266
37 G1 Z4.1 E-3.5259259259259266e-05
38
39 G1 F3000 X31.987 Y32.088997 E0
40 G1 Z1.1 F3.5259259259259266e-05

```

Figure 6: Example G-code.

References

- [1] Python Software Foundation. *Python Download*, 2020 (accessed April 27, 2020).
- [2] Anaconda Inc. *Anaconda Download*, 2020 (accessed April 27, 2020).
- [3] Julia Müller and Anna Jäkel. *Gel printer*, 2018 (accessed April 27, 2020).