Gel Printer Software HowTo

Jäkel, Anna Christina

April 2020

Contents

1 Brief Overview

The GitLab repository *Gel printer* hosted by the LRZ (`https://github.com/julia-mueller/bioprinter`) contains the complete source code written by Julia Müller and Anna Jäkel to produce G-code that can be executed by a gel loaded 3D printer.
This code enables the conversion of a svg file containing the structure that you like to print into machine readable G-code. It is further possible to specify some printing parameters, e.g. speed, feedrate, or volume that should be extruded. In figure 1 an example svg file and the resulting 3D object produced with this svg file are shown.

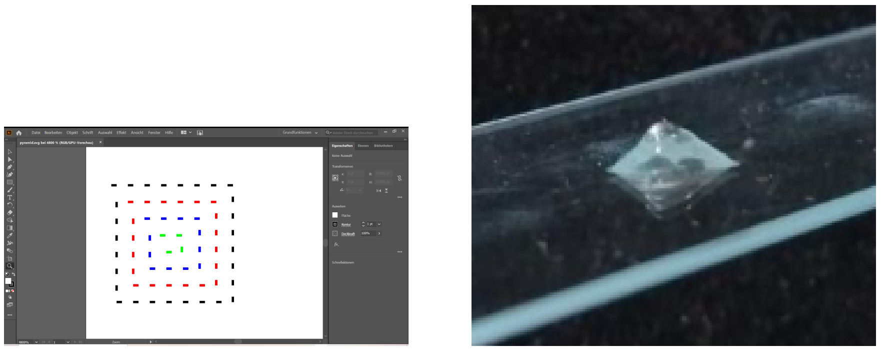


Figure 1: The svg file and the resulting print are shown.

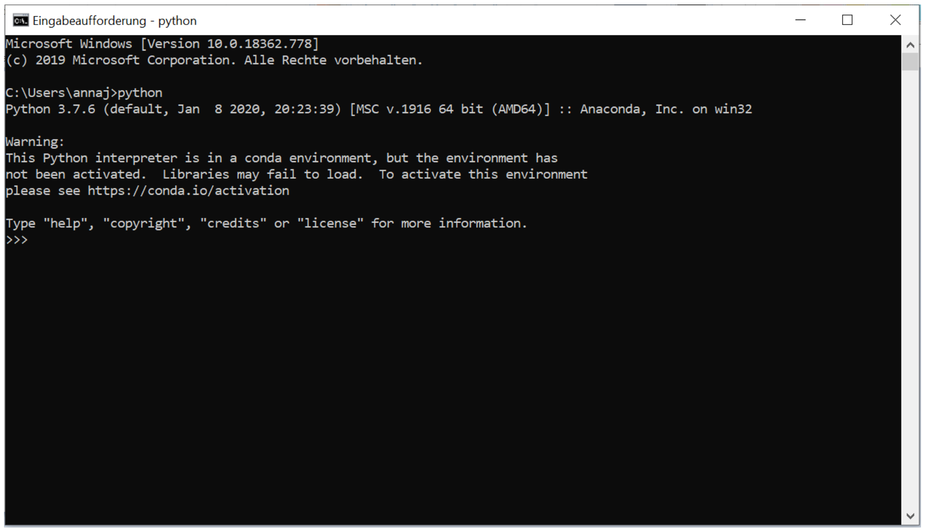
2 Software (Preparations)

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 jump to section

2.1 How to Install Python

Our code is written in Python 3.0. To use our code it is necessary to first download Python, which can be done on the official Python website `https://www.python.org/downloads/`. 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 if you use Python 2. Beside of that it is from now on suggested to use Python 3 since Python 2 is in EOL (End of Life) status.
After downloading the Python 3 installation file you have to follow the installation instructions of the installer which will open after you double clicked 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. You can run python code by typing "python *mycode.py*", where *mycode.py* is



```
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

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 Anaconda instead of the installation of Python 3.0 we mentioned above, because if any additional packages are required to compile certain code 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 (see section 3.1) of this Github directory to a location of your local computer or you use git to do that for you. We provide a short introduction to git in the pdf file *Instruction for GitLab* available here: `https://github.com/julia-mueller/bioprinter`.

3 G-code generator code

This section explains the input parameters and gives support on how to execute our code.

3.1 Description of the Files

To use the python code for converting svg files to G-code you need the following files:

- executor.py
- generator.py
- routinegernator.py
- constants.py

Also the following folders are necessary:

- parameterfiles : which contains your parameterfile
- gcodefiles : which contains your output G-code files
- svgfiles : which contains your svgfile

Only a few of these files are of interest for you. Those are explained in the following.
In the folder *parameterfiles* you have to save your parameterfiles in which you define all the parameters for the print. The parameters are explained in the next section. The parameterfile is the file you will call in the command line to generate your G-code. How this is done will be explained in section 3.4. In the folder *gcodefiles* you will find your generated G-code that you can use later, additional information about this is given in section 4. In the folder *svgfiles* you should save your created svg files. You have to make sure that the svg file is in a specific format. This is explained in details in section 3.2. The last file that is of importance for you is the *generator.py* file. This will be used to compile the code. How this is done is explained in section 3.4.

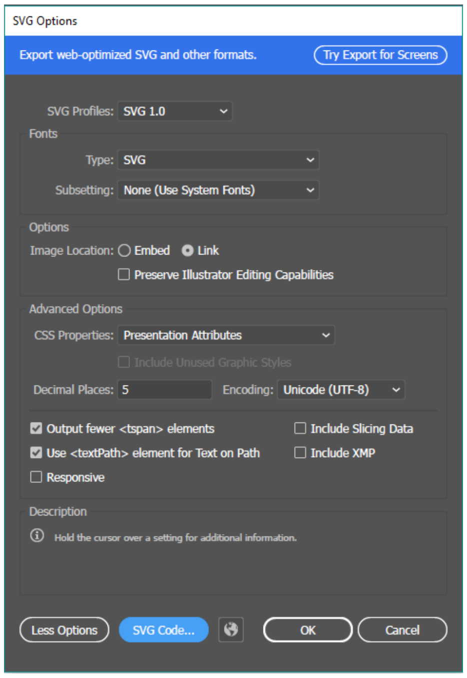
3.2 The SVG File

We recommend to use Illustrator for creating svg files, but any other vector graphic program works as well. We decided to print structures pointwise, each point is presented in the svgfile with a line of the length $0.238mm$. It is important to use a different width than 1 as linewidth, as this parameter will not be saved when it is 1 (our implementation needs this parameter). Usually the upper left corner is the origin of your drawing area. Design your structure in a mm scale and remember that the origin of the svg file will be the starting point of the printer. The starting point is defined to be 30mm in x and y direction from the lower left corner of the printplate of the ultimaker printplate.
With the usage of different colors it is possible to define different structures for each layer, as one parameter is the colorlist. The colorlist defines which color should be printed in each layer. It is also possible to assign different colors to different extruder, as our implementation allows several extruder. If you want to pause one extruder in a specific layer then you have to tell this extruder to print white in this specific layer. How exactly you do that is explained in the parameter section 3.3. You should save the svg file point- or pixelwise and with enough decimal places. The distances are then also read in as mm. So far it is only possible to read lines for our program, but we are working on polylines also. When saving the file with the new Illustrator version these steps are important during the saving process (shown in figure 3):

- draw in a px-defined new artboard
- ungroup everything
- the lines may not have an id
- export as (take care that the decimal places are sufficient! you can safe a .ai as backup before closing your file)

3.3 Description of the Paramters which could be changed

You will find an example parameterfile provided in the folder parameterfiles as mentioned before. Figure 4 shows how this looks like. In the following all the parameters available are listed and a short explanation is given:



SVG Options

Export web-optimized SVG and other formats.

SVG Profiles: SVG 1.0

Fonts

Type: SVG

Subsetting: None (Use System Fonts)

Options

Image Location: Embed [✓ Link]

☐ Preserve Illustrator Editing Capabilities

Advanced Options

CSS Properties: Presentation Attributes

☐ Include Unused Graphic Styles

Decimal Places: 5 Encoding: Unicode (UTF-8)

☑ Output fewer <tspan> elements

☐ Include Slicing Data

☑ Use <textPath> element for Text on Path

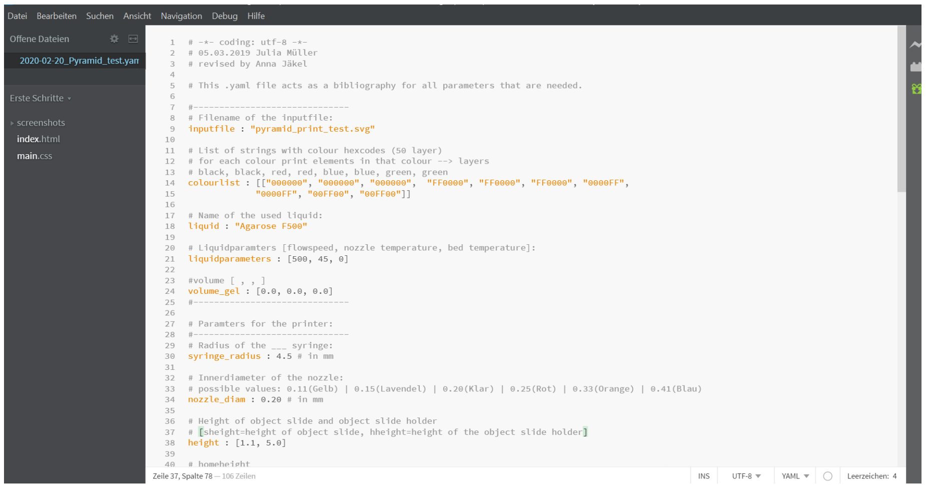
☐ Include XMP

☐ Responsive

Description

Hold the cursor over a setting for additional information.

Figure 3: Options for saving a svg file in illustrator.



Datei Bearbeiten Suchen Ansicht Navigation Debug Hilfe
Offene Dateien
2020-02-20_Pyramid_test.yam
Erste Schritte
screenshots
index.html
main.css
Zeile 37, Spalte 78 — 106 Zeilen
INS
UTF-8
YAML
Leerzeichen: 4

```yaml
# -*- coding: utf-8 -*-
# 05.03.2019 Julia Müller
# revised by Anna Jäkel

# This .yaml file acts as a bibliography for all parameters that are needed.

#------------------------------
# Filename of the inputfile:
inputfile : "pyramid_print_test.svg"

# List of strings with colour hexcodes (50 layer)
# for each colour print elements in that colour --> layers
# black, black, red, red, blue, blue, green, green
colourlist : [["000000", "000000", "000000",  "FF0000", "FF0000", "FF0000", "0000FF",
               "0000FF", "00FF00", "00FF00"]]

# Name of the used liquid:
liquid : "Agarose F500"

# Liquidparamters [flowspeed, nozzle temperature, bed temperature]:
liquidparameters : [500, 45, 0]

#volume [ , , ]
volume_gel : [0.0, 0.0, 0.0]
#------------------------------

# Paramters for the printer:
#------------------------------
# Radius of the ___ syringe:
syringe_radius : 4.5 # in mm

# Innerdiameter of the nozzle:
# possible values: 0.11(Gelb) | 0.15(Lavendel) | 0.20(Klar) | 0.25(Rot) | 0.33(Orange) | 0.41(Blau)
nozzle_diam : 0.20 # in mm

# Height of object slide and object slide holder
# [sheight=height of object slide, hheight=height of the object slide holder]
height : [1.1, 5.0]

# homeheight
```

Figure 4: Parameterfile

```
% Update these material parameters for each print:
```

```
inputfile: 'name_of_svg_file'
colourlist: [["color_for_extruder1_layer1", "
   color_for_extruder1_layer1", ...]["
   color_for_extruder2_layer1", "
   color_for_extruder2_layer2", ...]] (The colours need
   to be in Hexcode, the number of colours determines the
    number of layers which should be printed, the colour
   determines which parts of the svg should be printed)
nozzle_temp: 45 (nozzle temperature in °C, for our bioink
    this was 45°C)
bed_temp: 0 (bed temperature in °C, we used a bed at room
    temperature, so we set this to 0)
height: [height of the object slide, height of the object
    slide holder]
red_delta_z: This paramter determines how the height per
   layer is reduced (0.8, e.g.)
increase_volume_factor: 1.0 (if the volume should be
   doubled: 2.0, e.g.)

% Update these material parameters to optimize your print
    results:

scale_z: True or False, This paramter determines if the
   nozzle-sample distance should be changed
max_normal_layer: This paramter determines after how many
    layers the height should be reduced (8, e.g.)
dim_z_later: This parameter determines how the height
   sample-nozzle should be changed after a given number
   of layers (0.8, e.g.)
scale_z_squared: True or False,  scale the height
   quadratically
red_z_square: factor that decreases the quadratic part of
    the reduce-z-function (0.01, e.g.)
corner_adjustment: True or False, This parameter
   determines if the corner adjustment should happen or
   not

% Update these parameters for your own printer at first
   usage:

syringe_radius: radius of syringe in mm
nozzle_diam: 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: 1.5 (Area of the nozzle divided
     by the area of the syringe)
offset: offset between the extruder in mm (e.g. 50.0)
homeheight: in mm (this is the height of the mechanic or
    electric switch that sets z = 0)
slippage: value for the slippage of the printer (0.5, e.g
    .)
startdis: 30 (home position before print starts, 30mm in
    x and y direction from the (0.0,0.0) point)

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

volume_gel: [0.0, 0.0, 0.0]
begin_of_print: 0
z_safe_dis: This paramter determines the height added to
    z during movements (3, e.g.)
```

3.4 Compiling this Python Code

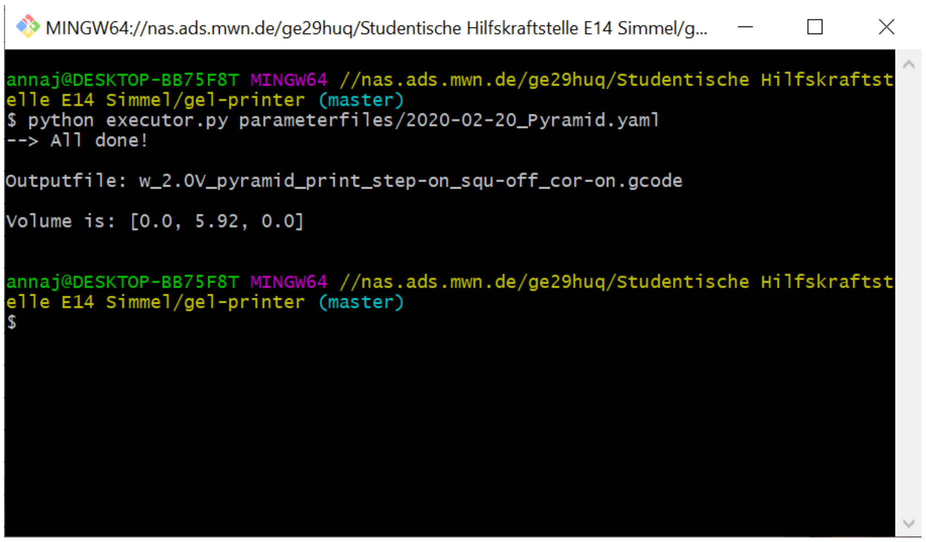
Open a command line and change to the folder where the python scripts and the above described folders are in. Then type in

```
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.

4 Working with the G-code

After you compiled the python code and succefully created a G-code file, you will find this G-code file in the folder *gcodefiles* in a subfolder with the name of the date of execution. The name of the G-code file has the format *nozzlediameter_svg-filename_date*. If you like to change how the filename is build, this can be done in the python script *generator.py* in the function *get_filename*. (But as this two parameters are the most important ones to choose the correct nozzle for printing and also print the correct figure we chose these parameters to build the filename.) We recommend to include at least the svg filename and the nozzle to your filename to facilitate your print.
TaDaa! You now can copy this G-code to a SD card or USB drive and print it with a gel laden 3D printer.


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)
$
```

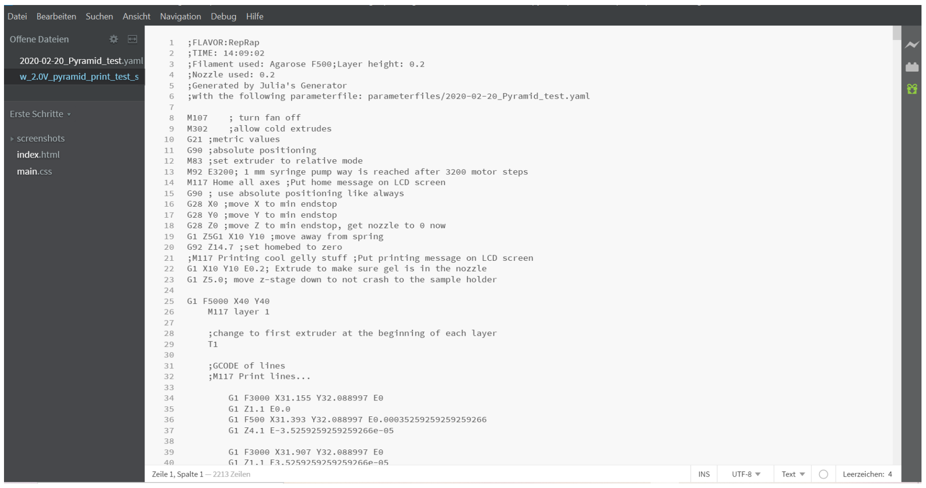
Figure 5: Compilation of a parameterfile.

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).



```
;FLAVOR:RepRap
;TIME: 14:09:02
;Filament used: Agarose F500;Layer height: 0.2
;Nozzle used: 0.2
;Generated by Julia's Generator
;with the following parameterfile: parameterfiles/2020-02-20_Pyramid_test.yaml

M107    ; turn fan off
M302    ;allow cold extrudes
G21 ;metric values
G90 ;absolute positioning
M83 ;set extruder to relative mode
M92 E3200; 1 mm syringe pump way is reached after 3200 motor steps
M117 Home all axes ;Put home message on LCD screen
G90 ; use absolute positioning like always
G28 X0 ;move X to min endstop
G28 Y0 ;move Y to min endstop
G28 Z0 ;move Z to min endstop, get nozzle to 0 now
G1 Z5G1 X10 Y10 ;move away from spring
G92 Z14.7 ;set homebed to zero
;M117 Printing cool gelly stuff ;Put printing message on LCD screen
G1 X10 Y10 E0.2; Extrude to make sure gel is in the nozzle
G1 Z5.0; move z-stage down to not crash to the sample holder

G1 F5000 X40 Y40
    M117 layer 1

    ;change to first extruder at the beginning of each layer
    T1

    ;GCODE of lines
    ;M117 Print lines...

        G1 F3000 X31.155 Y32.088997 E0
        G1 Z1.1 E0.0
        G1 F500 X31.393 Y32.088997 E0.00035259259259259266
        G1 Z4.1 E-3.5259259259259266e-05

        G1 F3000 X31.907 Y32.088997 E0
        G1 Z1.1 E3.5259259259259266e-05
```

Figure 6: Example G-code.