gcode_controls

March 30, 2024

1 GcodeControls adjust how a *design* is transformed into a 'gcode' *result*

designs are transformed into 'gcode' according to some default settings which can be overwritten with a GcodeControls object with the following attributes (all demonstrated in this notebook):

- `save_as` - used to save the gcode directly to a .gcode file
- `include_date` - append filename with date+time (default: True)
- `printer_name` - used to choose which printer the gcode should be formatted for
  - a selection of printers are built in, which is expected to be extended. documentation for adding new printers will be provided in the future
- `initialization_data` - used to change the initial print conditions/settings that are established before the steps in the ***design*** are evaluated

<this document is a jupyter notebook - if they're new to you, check out how they work: link, link, link>

run all cells in this notebook in order (keep pressing shift+enter)

[ ]:
```
import fullcontrol as fc
```

1.1 save to file

use the `save_as` attribute of a GcodeControls object to save the gcode to a file in the same directory as this notebook, with the filename appended by date and time

set `include_date` = False to remove date and time

[ ]:
```
steps = []
steps.append(fc.Point(x=30, y=30, z=0.2))
steps.append(fc.Point(x=60))
# option 1 (use built-in function):
gcode_controls = fc.GcodeControls(save_as='my_design') # filename includes␣
↪date+time
fc.transform(steps, 'gcode', gcode_controls)
gcode_controls = fc.GcodeControls(save_as='my_design', include_date = False) #␣
↪filename doesn't include date+time
fc.transform(steps, 'gcode', gcode_controls)
# option 2 (save gcode string to file manually):
gcode = fc.transform(steps, 'gcode')
```

```
open('my_file.gcode', 'w').write(gcode)
```

1.2 choose printer

change which printer to output gcode for with the 'printer_name' attribute

current options:

- 'generic' *(default)*
- 'ultimaker2plus'
- 'prusa_i3'
- 'ender_3'
- 'cr_10'
- 'bambulab_x1'
- 'toolchanger_T0'
- 'toolchanger_T1'
- 'toolchanger_T2'
- 'toolchanger_T3'
- 'custom'

the option 'generic' is default and outputs gcode with no start/end gcode except for the command M83, since omitting this command is a common source of error

the option 'custom' doesn't generate any start_gcode at all and allows custom starting procedures to be created as demonstrated later in this notebook

[ ]:
```
steps = []
steps.append(fc.Point(x=30, y=30, z=0.2))
steps.append(fc.Point(x=60))
gcode_controls = fc.GcodeControls(printer_name='toolchanger_T0',␣
 ↪save_as='my_design')
fc.transform(steps, 'gcode', gcode_controls)
```

1.3 change initial settings

the 'initialization_data' attribute is used to pass a python 'dictionary' capturing information about printing conditions/settings at the start of the printing process

currently, the dictionary can contain any of the aspects listed below

a description or the object type (defined in state objects notebook) is displayed next to each term along with default values - individual printers may over-ride these default values or they can be manually over-ridden by including them in the dictionary that is passed to the fc.transform() function when it generates gcode

- 'print_speed': 1000 - Printer(print_speed)
- 'travel_speed': 8000 - Printer(travel_speed)
- 'area_model': 'rectangle' - ExtrusionGeometry(area_model)
- 'extrusion_width': 0.4 - ExtrusionGeometry(width)
- 'extrusion_height': 0.2 - ExtrusionGeometry(height)
- 'nozzle_temp': 210 - Hotend(temp)

- 'bed_temp': 40 - Buildplate(temp)
- 'fan_percent': 100 - Fan(speed_percent)
- 'print_speed_percent': 100 - used in start_gcode for an M220 command
- 'material_flow_percent': 100 - used in start_gcode for an M221 command
- 'e_units': 'mm' - Extruder(units)
- 'relative_e': True - Extruder(relative_gcode)
- 'dia_feed': 1.75 - Extruder(dia_feed)
- 'primer': 'front_lines_then_y' - see later section about built-in primer options

default settings

[ ]:
```
steps = [fc.Point(x=30, y=30, z=0.2), fc.Point(x=60)]
print(fc.transform(steps, 'gcode'))
```

initial speed and fan as described above, the default printer is called 'generic' and outputs gcode with no start/end gcode except for the command M83. however, overriding an initial setting results in the appropriate gcode being added to start_gcode

[ ]:
```
steps = [fc.Point(x=30, y=30, z=0.2), fc.Point(x=60)]
initial_settings = {
    "print_speed": 2000,
    "travel_speed": 4000,
    "nozzle_temp": 280,
    "bed_temp": 80,
    "fan_percent": 40,
}
gcode_controls = fc.GcodeControls(initialization_data=initial_settings)
print(fc.transform(steps, 'gcode', gcode_controls))
```

extrusion width and parameters that affect E in gcode

[ ]:
```
steps = [fc.Point(x=30, y=30, z=0.2), fc.Point(x=60)]
initial_settings = {
    "extrusion_width": 0.8,
    "extrusion_height": 0.3,
    "e_units": "mm3",
    "relative_e": False,
    "dia_feed": 2.85,
}
gcode_controls = fc.GcodeControls(initialization_data=initial_settings)
print(fc.transform(steps, 'gcode', gcode_controls))
```

setting flow % and speed % these aspects change the over-ride values for speed % (gcode M220) and flow % (gcode M221). They don't change the values written for F terms and E terms in gcode. The printer display screen should show these values correctly during printing and allow them to be changed after the print has started

[ ]:
```
steps = [fc.Point(x=30, y=30, z=0.2), fc.Point(x=60)]
initial_settings = {
```

```
    "print_speed_percent": 100,
    "material_flow_percent": 100,
}
gcode_controls = fc.GcodeControls(initialization_data=initial_settings)
print(fc.transform(steps, 'gcode', gcode_controls))
```

primer some basic options to add a primer before your design begins printing are included in this release of FullControl. a good alternative to using a built-in primer, is to manually design a primer at the beginning of the list of steps in a ***design***. such a primer can be truly optimized for the individual design to ensure printing begins perfectly and to minimize the risk of first-layer defects. see an example of this below

current options for primers are:

- 'front_lines_then_x' - this involves printing some lines on the front of the bed before moving in the **x** direction to the start point of the ***design***
- 'front_lines_then_y' - similar to above except move in **y** direction
- 'front_lines_then_xy' - similar to above except move in diagonal **xy** direction
- 'x' - move from the position at the end of start_gcode to the start point of the ***design*** along the **x** direction (after a y-direction move)
- 'y' - similar to above except move in x first, then y to the start point
- 'xy' - print directly from the end of the start gcode to the start point
- 'travel' - travel from the end of the start gcode to the start point

[ ]:
```
steps = [fc.Point(x=30, y=30, z=0.2), fc.Point(x=60)]
gcode_controls = fc.GcodeControls(initialization_data={"primer":␣
 ↪"front_lines_then_xy"})
print(fc.transform(steps, 'gcode', gcode_controls))
```

custom primer an easy way to add a custom primer, is to include it at the beginning of the ***design***

set the gcode initialization data to have the 'travel' primer-type to quickly travel to the start point of the custom primer

the ***design*** in the following code cell is transformed to a 'plot' ***result*** rather than 'gcode' for ease of inspection

[ ]:
```
design_steps = [fc.polar_to_point(centre=fc.Point(x=0, y=0, z=i*0.005),␣
 ↪radius=10+10*(i%2), angle=i) for i in range(500)]
primer_steps = fc.spiralXY(fc.Point(x=0, y=0, z=0.2), 2, 8, 0, 4, 128)
steps = primer_steps + design_steps
steps.append(fc.PlotAnnotation(point=fc.Point(x=2, y=0, z=0.2), label='primer␣
 ↪start'))
steps.append(fc.PlotAnnotation(point=fc.Point(x=10, y=0, z=0.1), label='design␣
 ↪start'))
steps.append(fc.PlotAnnotation(point=fc.Point(x=0, y=0, z=10), label='internal␣
 ↪spiral-primer ends near the main-design start point'))
fc.transform(steps, 'plot')
```

1.4 custom printer template

add your own printer by updating the code in the following code cell, which uses the 'custom' printer-type and includes appropriate FullControl objects as the first few steps in the ***design***

the following commands generate gcode during initialization of the printer in FullControl, and therefore, it's advisable **not** to use them to avoid their associated gcode appearing before your starting procedure: - relative_e / nozzle_temp / bed_temp / fan_percent / print_speed_percent / material_flow_percent / primer

instead, these aspects should be controlled by the custom starting procedure at the start of your ***design***, including turning the extruder on at the appropriate time

future documentation will explain how to add you own printer to the library of printers in the python source code

[ ]:
```
# create the initialize procedure (i.e. start_gcode)
initial_settings = {
    "extrusion_width": 0.8,
    "extrusion_height": 0.3,
    "e_units": "mm3",
    "dia_feed": 2.85,
    "primer": "no_primer",
    "print_speed": 2000,
    "travel_speed": 4000
}
gcode_controls = fc.GcodeControls(printer_name='custom',␣
 ↪initialization_data=initial_settings)
starting_procedure_steps = []
starting_procedure_steps.append(fc.ManualGcode(text='\n; #####\n; #####␣
 ↪beginning of start procedure\n; #####'))
starting_procedure_steps.append(fc.ManualGcode(text='G28 ; home'))
starting_procedure_steps.append(fc.GcodeComment(text='heat bed 10 degrees too␣
 ↪hot'))
starting_procedure_steps.append(fc.Buildplate(temp=60, wait=True))
starting_procedure_steps.append(fc.GcodeComment(text='allow bed to cool to the␣
 ↪correct temp and heat up nozzle'))
starting_procedure_steps.append(fc.Hotend(temp=220, wait=False))
starting_procedure_steps.append(fc.Buildplate(temp=50, wait=True))
starting_procedure_steps.append(fc.Hotend(temp=220, wait=True))
starting_procedure_steps.append(fc.Fan(speed_percent=100))
starting_procedure_steps.append(fc.Extruder(relative_gcode=True))
starting_procedure_steps.append(fc.Point(x=10, y=10, z=0.4))
starting_procedure_steps.append(fc.ManualGcode(text='; #####\n; ##### end of␣
 ↪start procedure\n; #####\n'))

# create the design
design_steps = []
design_steps.append(fc.Point(x=0, y=0, z=0.2))
```

```
design_steps.append(fc.Extruder(on=True))
design_steps.append(fc.Point(x=10, y=0, z=0.2))

# combine start procedure and design to create the overall procedure
steps = starting_procedure_steps + design_steps
print(fc.transform(steps, 'gcode', gcode_controls))
```