

```

"""
    This file contains parameters, helpers, and setup to
    create a basic gcode generation algorithm from line segments.

    The main

    Inputs:
        lines: the line segments to be converted into gcode commands for
    extrusion
        nozzle_diameter: the diameter of the 3D printer's nozzle
        filament_diameter: the diameter of the 3d printing filament
        layer_height: the height of each layer in the print
        extrusion_width: the width of the extruded line from the printer
        travel_feed_rate: the speed at which the extruder moves in X and Y
        layer_change_feed_rate: the speed at which teh extruder moves when
            changing layers in the Z direction
        extrusion_feed_rate: the speed at which the extruder move when
    extruding

    Output:
        gcode_output: a string with gcode commands separate by new-lines"""

__author__ = "mriviera-cu"

import rhinoscriptsyntax as rs

import math

##### CONSTANTS BELOW #####

# GCODE COMMANDS
COMMAND_MOVE = "G1"

# GCODE PARAM NAMES

```

```

PARAM_X = "X"
PARAM_Y = "Y"
PARAM_Z = "Z"
PARAM_E = "E"
PARAM_F = "F"

# Separates commands
COMMAND_DELIMITER = "\n"

# Precision for converting floats to strings
E_VALUE_PRECISION = 5
XYZ_VALUE_PRECISION = 3

# Float equality precision
FLOAT_EQUALITY_PRECISION = 5

#### WARNING: DO NOT EDIT THE START GCODE! ####
start_gcode_lines = [";START GCODE",
    "M201 X1000 Y1000 Z200 E5000 ; sets maximum accelerations, mm/sec^2",
    "M203 X200 Y200 Z12 E120 ; sets maximum feedrates, mm / sec",
    "M204 S1250 T1250 ; sets acceleration (S) and retract acceleration (R),
mm/sec^2",
    "M205 X8.00 Y8.00 Z0.40 E4.50 ; sets the jerk limits, mm/sec",
    "M205 S0 T0 ; sets the minimum extruding and travel feed rate, mm/sec",
    ";TYPE:Custom",
    "M862.3 P \"MK3S\" ; printer model check",
    "M862.1 P0.4 ; nozzle diameter check",
    "M115 U3.13.3 ; tell printer latest fw version",
    "G90 ; use absolute coordinates",
    "M83 ; extruder relative mode",
    "M104 S215 ; set extruder temp",
    "M140 S60 ; set bed temp",
    "M190 S60 ; wait for bed temp",
    "M109 S215 ; wait for extruder temp",
    "G28 W ; home all without mesh bed level",
    "G80 X86.6877 Y64.7708 W85.2412 H59.1573 ; mesh bed levelling",

```

```

"G1 Z0.2 F720",
"G1 Y-3 F1000 ; go outside print area",
"G92 E0",
"G1 X60 E9 F1000 ; intro line",
"G1 X100 E12.5 F1000 ; intro line",
"G92 E0",
"M221 S95",
"; Don't change E values below. Excessive value can damage the
printer.",
"M907 E538 ; set extruder motor current",
"G21 ; set units to millimeters",
"G90 ; use absolute coordinates",
"M83 ; use relative distances for extrusion",
"M900 K0.05 ; Filament gcode LA 1.5",
"M900 K30 ; Filament gcode LA 1.0",
"M107"]

```

##### WARNING: DO NOT EDIT THE END GCODE! #####

```

end_gcode_lines = ["; END Gcode",
"M204 S1000",
"M107",
";TYPE:Custom",
"; Filament-specific end gcode",
"G1 Z180 F720 ; Move print head further up",
"G1 X0 Y200 F3600 ; park",
"G4 ; wait",
"M221 S100 ; reset flow",
"M900 K0 ; reset LA",
"M104 S0 ; turn off temperature",
"M140 S0 ; turn off heatbed",
"M107 ; turn off fan",
"M84 ; disable motors",
"M73 P100 R0 ; print progress done",
"M73 Q100 S0 ; print progress done"]

```

#####

```

# Converts a float (f) to a string with some precision of decimal places
# For example:
# Input: f=0.1234, precision=3
# Output: "0.123"
def float_to_string(f, precision=XYZ_VALUE_PRECISION):
    f = float(f)
    str_format = "{value:." + str(precision) + "f}"
    return str_format.format(value=f)

# Helper to convert the E value to the proper precision
def e_value_to_string(e):
    return float_to_string(e, E_VALUE_PRECISION)

# Helper to convert the XYZ value to the proper precision
def xyz_value_to_string(e):
    return float_to_string(e, XYZ_VALUE_PRECISION)

#####

# Helper function to compare floats in grasshopper/python due to float
precision errors
def are_floats_equal(f1, f2, epsilon=10**(-FLOAT_EQUALITY_PRECISION)):
    f1 *= 10**FLOAT_EQUALITY_PRECISION
    f2 *= 10**FLOAT_EQUALITY_PRECISION
    return math.fabs(f2 - f1) <= epsilon

# Helper function to compare if two points are equal (have the same
coordinates)
# by handling float precision comparisons
def is_same_pt(ptA, ptB):
    return are_floats_equal(ptA[0], ptB[0]) and are_floats_equal(ptA[1],
ptB[1]) and are_floats_equal(ptA[2], ptB[2])

#####

```

```

# creates a string consisting of a G1 move command and
# any associated parameters
def gcode_move(current_pos, next_pos, feed_rate=None,
should_extrude=False):
    # Start with "G1" as command
    move_command_str = COMMAND_MOVE

    # Compare X positions
    if (not are_floats_equal(current_pos[0], next_pos[0])):
        # we have a different x position so add it as a parameter to the
command
        x_value = float_to_string(next_pos[0],
precision=XYZ_VALUE_PRECISION)
        # Add X<x_value> to move string, e.g., X100.00
        move_command_str += " " + PARAM_X + x_value

    # Compare Y positions
    if (not are_floats_equal(current_pos[1], next_pos[1])):
        # we have a different y position so add the new position as a
parameter
        y_value = float_to_string(next_pos[1],
precision=XYZ_VALUE_PRECISION)
        # Add Y<y_value> to move string, e.g., Y100.00
        move_command_str += " " + PARAM_Y + y_value

    # Compare Z position
    if (not are_floats_equal(current_pos[2], next_pos[2])):
        # we have a different z position so add the new position as a
parameter
        z_value = float_to_string(next_pos[2],
precision=XYZ_VALUE_PRECISION)
        # Add Z<z_value> to move string, e.g., Z100.00
        move_command_str += " " + PARAM_Z + z_value

    # [TODO]: handle "should_extrude" == true by determining the proper
amount to

```

```

    # extrude using the capsule model, then append the E parameter and
value
    # to the move command.
    # NOTE: YOUR FLOAT PRECISION MUST MATCH E_VALUE_PRECISION

    if (should_extrude == True):
        distance = next_pos[0] - current_pos[0]
        V_in = math.pi * (filament_diameter / 2) ** 2
        V_out = ((extrusion_width - layer_height) * layer_height + math.pi
* (layer_height / 2) ** 2) * distance
        L = V_out / V_in
        E = float_to_string(L, precision = E_VALUE_PRECISION)
        move_command_str += " " + PARAM_E + E

    # See if we have a feedrate to use, and handle it differently than
other
    # parameters as it is an integer value
    if (feed_rate is not None):
        # feed rate is an int
        feed_rate_value = round(feed_rate)
        # Add F<feed_rate_value> to move string, e.g., F2000
        move_command_str += " " + PARAM_F + str(feed_rate_value)

    # Remove excess whitespace on ends
    move_command_str = move_command_str.strip(" ")
    return move_command_str

#####
#####
#####

''' Here's the main method of the script that uses the helper methods
above '''

def generate_gcode():

```

```

# [TODO]: Implement the algorithm to generate gcode for each layer by
# first to moveing to the layer height, then moving to each line
segment.

# Once at a line segment, you should move and extrude along it,
# then move (travel) to the next line until there are no lines left
# For each of these movements, you should append the command to
# the list: `all_move_commands`
current_position = [0, 0, 0]          # start extruder at the origin
all_move_commands = []                # list to hold for all the move
commands

for i in range(0, len(lines)):
    # Get pts of the line segment
    line = lines[i]
    line_start_position = line[0]
    line_end_position = line[1]

    # [TODO]: Handle moving to the next layer (Z Position)
    # NOTE- ALL Z MOVEMENTS SHOULD:
    # 1) BE INDEPENDENT MOVES(e.g., G1 Z# and not move with other
positions like XYE)
    # 2) USE THE `layer_change_feedrate`
    # 3) BE IN INCREASING ORDER

    if not are_floats_equal(current_position[2],
line_start_position[2]):
        move_layer = COMMAND_MOVE + " " + PARAM_Z +
xyz_value_to_string(line_start_position[2])
        all_move_commands.append(move_layer)
        current_position[2] = line_start_position[2]

    # Now if our current_position is not the start of our line segment
    # we need to move (travel) to the line segment's starting point
    if not is_same_pt(current_position, line_start_position):
        # [Move to the the line_start_position

```

```

        move_to_line_start_command = gcode_move(current_position,
line_start_position, feed_rate = travel_feed_rate)
        # A ppend command
        all_move_commands.append(move_to_line_start_command)
        current_position = line_start_position

        # [TODO]: Once our extruder is at the start of the line, create a
        # command to move AND extrude along
        # the line segment using `extrusion_feed_rate`
        line_length = math.sqrt((line_end_position[0] -
line_start_position[0]) ** 2 + (line_end_position[1] -
line_start_position[1]) ** 2 + (line_end_position[2] -
line_start_position[2]) ** 2)
        extrusion_volume = line_length * extrusion_width * layer_height
        move_and_extrude = gcode_move(line_start_position,
line_end_position, feed_rate = extrusion_feed_rate, should_extrude = True)
        all_move_commands.append(move_and_extrude)

        # [TODO]: Append the move command across the line segment
        move_command_line = gcode_move(line_start_position,
line_end_position, feed_rate=travel_feed_rate)
        all_move_commands.append(move_command_line)

        # [TODO]: Update the current position of our extruder to be at the
end of the line
        current_position = line_end_position
        current_position = [0, 0, 0]

# End of for-loop above -- now create the full set of commands

# [TODO]: Once you have all of the move commands stored as a list in
# `all_move_commands`, combine the `start_gcode`, `all_move_commands`,
and `end_gcode`
# into one list called `gcode_lines`
gcode_lines = start_gcode_lines + all_move_commands + end_gcode_lines

```



```
# --- DO NOT EDIT BELOW ----
# The takes the combined gcode_lines list and creates a string
containing each line
# separated by a COMMAND_DELIMITER (the new-line character), and sets
it
# to the `gcode_output` variable of this component
output = COMMAND_DELIMITER.join(gcode_lines)

return output

''' RUN THE MAIN FUNCITON ABOVE - DO NOT EDIT '''
# this sets the gcode commands to be the the `gcode_output` variable of
this grasshopper component
gcode_output = generate_gcode()
```