The gcodepreview OpenSCAD library*

Author: William F. Adams
willadams at aol dot com

2024/04/12

Abstract

The gcodepreview library allows using OpenSCAD to move a tool in lines and output dxf and G-code files so as to work as a CAD/CAM program for CNC.

Contents

1 readme.md

```
1 ⟨rdm⟩ # gcodepreview
2 ⟨rdm⟩
3 ⟨rdm⟩ OpenSCAD library for moving a tool in lines and arcs so as to model how a part would be cut
4 ⟨rdm⟩ using G-Code, so as to allow OpenSCAD to function as a compleat CAD/CAM solution for
5 ⟨rdm⟩ subtractive CNC (mills and routers).
6 ⟨rdm⟩
7 ⟨rdm⟩ ![OpenSCAD Cut Joinery Module](https://raw.githubusercontent.com/WillAdams/gcodepreview/mai
8 ⟨rdm⟩
9 ⟨rdm⟩ Updated to make use of Python in OpenSCAD:
10 ⟨rdm⟩
11 ⟨rdm⟩ http://www.guenther-sohler.net/openscad/
12 ⟨rdm⟩
13 ⟨rdm⟩ (previous versions had used RapCAD)
14 ⟨rdm⟩
15 ⟨rdm⟩ A BlockSCAD file for the main modules is available at:
16 ⟨rdm⟩
```

*This file (`gcodepreview.dtx`) has version number v0.2, last revised 2024/04/12.

```
⟨rdm⟩ https://www.blockscad3d.com/community/projects/1244473
⟨rdm⟩
⟨rdm⟩ The project is discussed at:
⟨rdm⟩
⟨rdm⟩ https://forum.makerforums.info/t/g-code-preview-using-openscad-rapcad/85729
⟨rdm⟩
⟨rdm⟩ and
⟨rdm⟩
⟨rdm⟩ https://forum.makerforums.info/t/openscad-and-python-looking-to-finally-be-resolv
⟨rdm⟩
⟨rdm⟩ and
⟨rdm⟩
⟨rdm⟩ https://willadams.gitbook.io/design-into-3d/programming
⟨rdm⟩
⟨rdm⟩ Usage is:
⟨rdm⟩
⟨rdm⟩ Place the file in C:\Users\\\~\Documents\OpenSCAD\libraries (C:\Users\\\~\Documen
⟨rdm⟩ deprecated since RapCAD is not longer needed since Python is now used for writing
⟨rdm⟩
⟨rdm⟩ (While it was updated for use w/ RapCAD, so as to take advantage of the writeln c
⟨rdm⟩ it was possible to write that in Python)
⟨rdm⟩
⟨rdm⟩     use <gcodepreview.py>;
⟨rdm⟩     use <pygcodepreview.scad>;
⟨rdm⟩     include <gcodepreview.scad>;
⟨rdm⟩
⟨rdm⟩ Note that it is necessary to use the first two files (this allows loading
⟨rdm⟩ the Python commands and then wrapping them in OpenSCAD commands) and then
⟨rdm⟩ include the last file (which allows using OpenSCAD variables to selectively
⟨rdm⟩ implement the Python commands via their being wrapped in OpenSCAD modules)
⟨rdm⟩
⟨rdm⟩ and define variables which match the project and then use commands such as:
⟨rdm⟩
⟨rdm⟩     opengcodefile(Gcode_filename);
⟨rdm⟩     opendxffile(DXF_filename);
⟨rdm⟩
⟨rdm⟩     difference() {
⟨rdm⟩         setupstock(stocklength, stockwidth, stockthickness, zeroheight, stockorig
⟨rdm⟩
⟨rdm⟩     movetosafez();
⟨rdm⟩
⟨rdm⟩     toolchange(squaretoolno,speed * square_ratio);
⟨rdm⟩
⟨rdm⟩     begintoolpath(0,0,0.25);
⟨rdm⟩     beginpolyline(0,0,0.25);
⟨rdm⟩
⟨rdm⟩     cutoneaxis_setfeed("Z",-1,plunge*square_ratio);
⟨rdm⟩     addpolyline(stocklength/2,stockwidth/2,-stockthickness);
⟨rdm⟩
⟨rdm⟩     cutwithfeed(stocklength/2,stockwidth/2,-stockthickness,feed);
⟨rdm⟩
⟨rdm⟩     endtoolpath();
⟨rdm⟩     endpolyline();
⟨rdm⟩
```

```
71 ⟨rdm⟩     }
72 ⟨rdm⟩
73 ⟨rdm⟩     closegcodefile();
74 ⟨rdm⟩     closedxffile();
75 ⟨rdm⟩
76 ⟨rdm⟩ Tool numbers match those of tooling sold by Carbide 3D (ob. discl., I work for them).
77 ⟨rdm⟩ Comments are included in the G-code to match those expected by CutViewer.
78 ⟨rdm⟩
79 ⟨rdm⟩ A complete example file is: gcodepreview_template.scad another is
80 ⟨rdm⟩ openscad_gcodepreview_cutjoinery.tres.scad which is made from an
81 ⟨rdm⟩ OpenSCAD Graph Editor file:
82 ⟨rdm⟩
83 ⟨rdm⟩ ![OpenSCAD Graph Editor Cut Joinery File](https://raw.githubusercontent.com/WillAdams/gcodep
84 ⟨rdm⟩
85 ⟨rdm⟩ Version 0.1 supports setting up stock, origin, rapid positioning, making cuts,
86 ⟨rdm⟩ and writing out matching G-code, and creating a DXF with polylines.
87 ⟨rdm⟩
88 ⟨rdm⟩ Added features since initial upload:
89 ⟨rdm⟩
90 ⟨rdm⟩  - endpolyline(); --- this command allows ending one polyline so as to allow multiple lines
91 ⟨rdm⟩  - separate dxf files are written out for each tool where tool is ball/square/V and small/la
92 ⟨rdm⟩  - re-writing as a Literate Program using the LaTeX package docmfp (begun 4/12/24)
93 ⟨rdm⟩
94 ⟨rdm⟩ Not quite working feature:
95 ⟨rdm⟩
96 ⟨rdm⟩  - exporting SVGs --- these are written out upside down due to coordinate differences betwe
97 ⟨rdm⟩
98 ⟨rdm⟩ Possible future improvements:
99 ⟨rdm⟩
100 ⟨rdm⟩  - G-code: support for G2/G3 arcs
101 ⟨rdm⟩  - DXF support for curves and the 3rd dimension
102 ⟨rdm⟩  - G-code: import external tool libraries and feeds and speeds from JSON or CSV files --- no
103 ⟨rdm⟩  - support for additional tooling shapes such as dovetail tools, or roundover tooling
104 ⟨rdm⟩  - general coding improvements --- current coding style is quite prosaic
105 ⟨rdm⟩  - generalized modules for cutting out various shapes/geometries --- a current one is to cut
106 %
```

2 gcodepreview

As noted above, this library works by using Python code as a back-end so as to persistently store and access variables, and to write out files. Doing so requires a total of three files:

- A Python file: gcodepreview.py (gcpy)
- An OpenSCAD file: gcodepreview.scad (gcpscad)
- An OpenSCAD file which connects the other two files: pygcodepreview.scad (pyscad)

Each file will begin with a suitable comment indicating the file type:

```
107 ⟨gcpy⟩ #!/usr/bin/env python
108 ⟨gcpy⟩
```

```
⟨pyscad⟩ //!OpenSCAD
⟨pyscad⟩
```

```
⟨gcpscad⟩ //!OpenSCAD
⟨gcpscad⟩
⟨gcpscad⟩ //gcodepreview 0.1
⟨gcpscad⟩ //
⟨gcpscad⟩ //used via use <gcodepreview.py>;
⟨gcpscad⟩ //         use <pygcodepreview.scad>;
⟨gcpscad⟩ //         include <gcodepreview.scad>;
⟨gcpscad⟩ //
```

writeln The original implementation in RapSCAD used a command `writeln` — fortunately, this command is easily re-created in Python:

```
⟨gcpy⟩ def writeln(*arguments):
⟨gcpy⟩     line_to_write = ""
⟨gcpy⟩     for element in arguments:
⟨gcpy⟩         line_to_write += element
⟨gcpy⟩     f.write(line_to_write)
⟨gcpy⟩     f.write("\n")
```

which command will accept a series of arguments and then write them out to a file object.

2.1 Position and Variables

In modeling the machine motion and G-code it will be necessary to have the machine track several variables. This will be done using paired functions (which will return the matching variable) and a matching (global) variable, as well as additional functions for setting the matching variable.

The first such variables are for XYZ position:

mpx • `mpx`

mpy • `mpy`

mpz • `mpz`

It will further be necessary to have a variable for the current tool:

currenttool • `currenttool`

For each command it will be necessary to implement an appropriate aspect in each file. The Python file wil manage the Python variables and handle things which can only be done in Python, while there will be two OpenSCAD files as noted above, one which calls the Python code (thjs will be `use`d), while the other will be `include`d and will be able to access and use OpenSCAD variables, as well as implement Customizer options.

The first such routine will be appropriately enough, to set up the stock, and psetupstock perform other initializations.

```
⟨gcpy⟩ def psetupstock(stocklength, stockwidth, stockthickness, zeroheight, stockorigin
⟨gcpy⟩     global mpx
⟨gcpy⟩     mpx = float(0)
⟨gcpy⟩     global mpy
```

```
⟨gcpy⟩     mpy = float(0)
⟨gcpy⟩     global mpz
⟨gcpy⟩     mpz = float(0)
⟨gcpy⟩     global currenttool
⟨gcpy⟩     currenttool = 102
⟨gcpy⟩
```

osetupstock

```
⟨pyscad⟩ module osetupstock(stocklength, stockwidth, stockthickness, zeroheight, stockorigin) {
⟨pyscad⟩     psetupstock(stocklength, stockwidth, stockthickness, zeroheight, stockorigin);
⟨pyscad⟩ }
⟨pyscad⟩
```

setupstock

```
⟨gcpscad⟩ module setupstock(stocklength, stockwidth, stockthickness, zeroheight, stockorigin) {
⟨gcpscad⟩   osetupstock(stocklength, stockwidth, stockthickness, zeroheight, stockorigin);
⟨gcpscad⟩ //initialize default tool and XYZ origin
⟨gcpscad⟩   osettool(102);
⟨gcpscad⟩   oset(0,0,0);
⟨gcpscad⟩   if (zeroheight == "Top") {
⟨gcpscad⟩     if (stockorigin == "Lower-Left") {
⟨gcpscad⟩     translate([0, 0, (-stockthickness)]){
⟨gcpscad⟩     cube([stocklength, stockwidth, stockthickness], center=false);
⟨gcpscad⟩ if (generategcode == true) {
⟨gcpscad⟩ // owriteone("(setupstock)");
⟨gcpscad⟩ owritethree("(stockMin:0.00mm, 0.00mm, -",str(stockthickness),"mm)");
⟨gcpscad⟩ owritefive("(stockMax:",str(stocklength),"mm, ",str(stockwidth),"mm, 0.00mm)");
⟨gcpscad⟩     owritenine("(STOCK/BLOCK, ",str(stocklength),", ",str(stockwidth),", ",str(stockthic
⟨gcpscad⟩ }
⟨gcpscad⟩ }
⟨gcpscad⟩ }
⟨gcpscad⟩      else if (stockorigin == "Center-Left") {
⟨gcpscad⟩     translate([0, (-stockwidth / 2), -stockthickness]){
⟨gcpscad⟩       cube([stocklength, stockwidth, stockthickness], center=false);
⟨gcpscad⟩     if (generategcode == true) {
⟨gcpscad⟩ // owriteone("(setupstock)");
⟨gcpscad⟩ owritefive("(stockMin:0.00mm, -",str(stockwidth/2),"mm, -",str(stockthickness),"mm)");
⟨gcpscad⟩ owritefive("(stockMax:",str(stocklength),"mm, ",str(stockwidth/2),"mm, 0.00mm)");
⟨gcpscad⟩     owriteeleven("(STOCK/BLOCK, ",str(stocklength),", ",str(stockwidth),", ",str(stockth
⟨gcpscad⟩     }
⟨gcpscad⟩     }
⟨gcpscad⟩     } else if (stockorigin == "Top-Left") {
⟨gcpscad⟩     translate([0, (-stockwidth), -stockthickness]){
⟨gcpscad⟩       cube([stocklength, stockwidth, stockthickness], center=false);
⟨gcpscad⟩ if (generategcode == true) {
⟨gcpscad⟩ // owriteone("(setupstock)");
⟨gcpscad⟩ owritefive("(stockMin:0.00mm, -",str(stockwidth),"mm, -",str(stockthickness),"mm)");
⟨gcpscad⟩ owritethree("(stockMax:",str(stocklength),"mm, 0.00mm, 0.00mm)");
⟨gcpscad⟩ owriteeleven("(STOCK/BLOCK, ",str(stocklength),", ",str(stockwidth),", ",str(stockthickn
⟨gcpscad⟩     }
⟨gcpscad⟩     }
⟨gcpscad⟩     }
⟨gcpscad⟩  else if (stockorigin == "Center") {
⟨gcpscad⟩ //owritecomment("Center");
⟨gcpscad⟩     translate([(-stocklength / 2), (-stockwidth / 2), -stockthickness]){
```

```
⟨gcpscad⟩       cube([stocklength, stockwidth, stockthickness], center=false);
⟨gcpscad⟩ if (generategcode == true) {
⟨gcpscad⟩ // owriteone("(setupstock)");
⟨gcpscad⟩ owriteseven("(stockMin: -",str(stocklength/2),", -",str(stockwidth/2),"mm, -"
⟨gcpscad⟩ owritefive("(stockMax:",str(stocklength/2),"mm, ",str(stockwidth/2),"mm, 0.00
⟨gcpscad⟩ owritethirteen("(STOCK/BLOCK, ",str(stocklength),", ",str(stockwidth),", ",st
⟨gcpscad⟩ }
⟨gcpscad⟩ }
⟨gcpscad⟩ }
⟨gcpscad⟩ } else if (zeroheight == "Bottom") {
⟨gcpscad⟩ //owritecomment("Bottom");
⟨gcpscad⟩     if (stockorigin == "Lower-Left") {
⟨gcpscad⟩     cube([stocklength, stockwidth, stockthickness], center=false);
⟨gcpscad⟩ if (generategcode == true) {
⟨gcpscad⟩ // owriteone("(setupstock)");
⟨gcpscad⟩ owriteone("(stockMin:0.00mm, 0.00mm, 0.00mm)");
⟨gcpscad⟩ owriteseven("(stockMax:",str(stocklength),"mm, ",str(stockwidth),"mm, ",str(st
⟨gcpscad⟩ owriteseven("(STOCK/BLOCK, ",str(stocklength),", ",str(stockwidth),", ",str(st
⟨gcpscad⟩     }
⟨gcpscad⟩ } else if (stockorigin == "Center-Left") {
⟨gcpscad⟩     translate([0, (-stockwidth / 2), 0]){
⟨gcpscad⟩       cube([stocklength, stockwidth, stockthickness], center=false);
⟨gcpscad⟩ if (generategcode == true) {
⟨gcpscad⟩ // owriteone("(setupstock)");
⟨gcpscad⟩ owritethree("(stockMin:0.00mm, -",str(stockwidth/2),"mm, 0.00mm)");
⟨gcpscad⟩ owriteseven("(stockMax:",str(stocklength),"mm, ",str(stockwidth/2),"mm, ",str
⟨gcpscad⟩ owritenine("(STOCK/BLOCK, ",str(stocklength),", ",str(stockwidth),", ",str(sto
⟨gcpscad⟩     }
⟨gcpscad⟩     }
⟨gcpscad⟩  } else if (stockorigin == "Top-Left") {
⟨gcpscad⟩     translate([0, (-stockwidth), 0]){
⟨gcpscad⟩       cube([stocklength, stockwidth, stockthickness], center=false);
⟨gcpscad⟩     }
⟨gcpscad⟩ if (generategcode == true) {
⟨gcpscad⟩ // owriteone("(setupstock)");
⟨gcpscad⟩ owritethree("(stockMin:0.00mm, -",str(stockwidth),"mm, 0.00mm)");
⟨gcpscad⟩ owritefive("(stockMax:",str(stocklength),"mm, 0.00mm, ",str(stockthickness),"
⟨gcpscad⟩ owritenine("(STOCK/BLOCK, ",str(stocklength),", ",str(stockwidth),", ",str(sto
⟨gcpscad⟩ }
⟨gcpscad⟩ } else if (stockorigin == "Center") {
⟨gcpscad⟩     translate([(-stocklength / 2), (-stockwidth / 2), 0]){
⟨gcpscad⟩       cube([stocklength, stockwidth, stockthickness], center=false);
⟨gcpscad⟩     }
⟨gcpscad⟩ if (generategcode == true) {
⟨gcpscad⟩ // owriteone("(setupstock)");
⟨gcpscad⟩ owritefive("(stockMin:-",str(stocklength/2),", -",str(stockwidth/2),"mm, 0.00
⟨gcpscad⟩ owriteseven("(stockMax:",str(stocklength/2),"mm, ",str(stockwidth/2),"mm, ",st
⟨gcpscad⟩ owriteeleven("(STOCK/BLOCK, ",str(stocklength),", ",str(stockwidth),", ",str(
⟨gcpscad⟩ }
⟨gcpscad⟩ }
⟨gcpscad⟩ }
⟨gcpscad⟩ if (generategcode == true) {
⟨gcpscad⟩  owriteone("G90");
⟨gcpscad⟩  owriteone("G21");
```

```
⟨gcpscad⟩ // owriteone("(Move to safe Z to avoid workholding)");
⟨gcpscad⟩ // owriteone("G53G0Z-5.000");
⟨gcpscad⟩ }
⟨gcpscad⟩ //owritecomment("ENDSETUP");
⟨gcpscad⟩ }
⟨gcpscad⟩
```

xpos
ypos
zpos

It will be necessary to have Python functions which return the current values of the machine position in Cartesian coordinates:

```
⟨gcpy⟩ def xpos():
⟨gcpy⟩     global mpx
⟨gcpy⟩     return mpx
⟨gcpy⟩
⟨gcpy⟩ def ypos():
⟨gcpy⟩     global mpy
⟨gcpy⟩     return mpy
⟨gcpy⟩
⟨gcpy⟩ def zpos():
⟨gcpy⟩     global mpz
⟨gcpy⟩     return mpz
⟨gcpy⟩
```

psetxpos
psetypos
psetzpos

and in turn, functions which set the positions:

```
⟨gcpy⟩ def psetxpos(newxpos):
⟨gcpy⟩     global mpx
⟨gcpy⟩     mpx = newxpos
⟨gcpy⟩
⟨gcpy⟩ def psetypos(newypos):
⟨gcpy⟩     global mpy
⟨gcpy⟩     mpy = newypos
⟨gcpy⟩
⟨gcpy⟩ def psetzpos(newzpos):
⟨gcpy⟩     global mpz
⟨gcpy⟩     mpz = newzpos
⟨gcpy⟩
```

getxpos
getypos
getzpos
setxpos
setypos
setzpos

and as noted above, there will need to be matching OpenSCAD versions:

```
⟨pyscad⟩ function getxpos() = xpos();
⟨pyscad⟩ function getypos() = ypos();
⟨pyscad⟩ function getzpos() = zpos();
⟨pyscad⟩
⟨pyscad⟩ module setxpos(newxpos) {
⟨pyscad⟩ psetxpos(newxpos);
⟨pyscad⟩ }
⟨pyscad⟩
⟨pyscad⟩ module setypos(newypos) {
⟨pyscad⟩ psetypos(newypos);
⟨pyscad⟩ }
⟨pyscad⟩
⟨pyscad⟩ module setzpos(newzpos) {
⟨pyscad⟩ psetzpos(newzpos);
⟨pyscad⟩ }
⟨pyscad⟩
```

oset

```
⟨gcpscad⟩ module oset(ex, ey, ez) {
⟨gcpscad⟩ setxpos(ex);
⟨gcpscad⟩ setypos(ey);
⟨gcpscad⟩ setzpos(ez);
⟨gcpscad⟩ }
⟨gcpscad⟩
```

2.2 Tools and Changes

Similarly Python functions and variables will be used to track and set and return the current tool:

psettool
pcurrenttool

```
⟨gcpy⟩ def psettool(tn):
⟨gcpy⟩     global currenttool
⟨gcpy⟩     currenttool = tn
⟨gcpy⟩
⟨gcpy⟩ def pcurrent_tool():
⟨gcpy⟩     global currenttool
⟨gcpy⟩     return currenttool
⟨gcpy⟩
```

osettool
currenttool

and matching OpenSCAD modules set and return the current tool:

```
⟨pyscad⟩ module osettool(tn){
⟨pyscad⟩ psettool(tn);}
⟨pyscad⟩
⟨pyscad⟩ function current_tool() = pcurrent_tool();
⟨pyscad⟩
```

toolchange

and apply the appropriate commands for a `toolchange`.

```
⟨gcpscad⟩ module toolchange(tool_number,speed) {
⟨gcpscad⟩    osettool(tool_number);
⟨gcpscad⟩ if (generategcode == true) {
⟨gcpscad⟩  writecomment("Toolpath");
⟨gcpscad⟩  owriteone("M05");
⟨gcpscad⟩ // writecomment("Move to safe Z to avoid workholding");
⟨gcpscad⟩ // owriteone("G53G0Z-5.000");
⟨gcpscad⟩ //    writecomment("Begin toolpath");
⟨gcpscad⟩   if (tool_number == 201) {
⟨gcpscad⟩  writecomment("TOOL/MILL,6.35, 0.00, 0.00, 0.00");
⟨gcpscad⟩   } else if (tool_number == 202) {
⟨gcpscad⟩  writecomment("TOOL/MILL,6.35, 3.17, 0.00, 0.00");
⟨gcpscad⟩   } else if (tool_number == 102) {
⟨gcpscad⟩  writecomment("TOOL/MILL,3.17, 0.00, 0.00, 0.00");
⟨gcpscad⟩   } else if (tool_number == 101) {
⟨gcpscad⟩  writecomment("TOOL/MILL,3.17, 1.58, 0.00, 0.00");
⟨gcpscad⟩   } else if (tool_number == 301) {
⟨gcpscad⟩  writecomment("TOOL/MILL,0.03, 0.00, 6.35, 45.00");
⟨gcpscad⟩   } else if (tool_number == 302) {
⟨gcpscad⟩  writecommment("TOOL/MILL,0.03, 0.00, 10.998, 30.00");
⟨gcpscad⟩   } else if (tool_number == 390) {
⟨gcpscad⟩  writecomment("TOOL/MILL,0.03, 0.00, 1.5875, 45.00");
⟨gcpscad⟩   }
⟨gcpscad⟩     select_tool(tool_number);
⟨gcpscad⟩  owritetwo("M6T",str(tool_number));
⟨gcpscad⟩  owritetwo("M03S",str(speed));
```

```
⟨gcpscad⟩ }
⟨gcpscad⟩ }
⟨gcpscad⟩
```

selecttool
tool_number

There must also be a module for selecting tools: `select_tool`:

```
⟨gcpscad⟩ module select_tool(tool_number) {
⟨gcpscad⟩ //echo(tool_number);
⟨gcpscad⟩   if (tool_number == 201) {
⟨gcpscad⟩     gcp_endmill_square(6.35, 19.05);
⟨gcpscad⟩   } else if (tool_number == 202) {
⟨gcpscad⟩     gcp_endmill_ball(6.35, 19.05);
⟨gcpscad⟩   } else if (tool_number == 102) {
⟨gcpscad⟩     gcp_endmill_square(3.175, 19.05);
⟨gcpscad⟩   } else if (tool_number == 101) {
⟨gcpscad⟩     gcp_endmill_ball(3.175, 19.05);
⟨gcpscad⟩   } else if (tool_number == 301) {
⟨gcpscad⟩     gcp_endmill_v(90, 12.7);
⟨gcpscad⟩   } else if (tool_number == 302) {
⟨gcpscad⟩     gcp_endmill_v(60, 12.7);
⟨gcpscad⟩   } else if (tool_number == 390) {
⟨gcpscad⟩     gcp_endmill_v(90, 3.175);
⟨gcpscad⟩   }
⟨gcpscad⟩ }
⟨gcpscad⟩
```

Each tool must be modeled in 3D using an OpenSCAD module:

gcp_endmill_square

```
⟨gcpscad⟩ module gcp_endmill_square(es_diameter, es_flute_length) {
⟨gcpscad⟩   cylinder(r1=(es_diameter / 2), r2=(es_diameter / 2), h=es_flute_length, center=false);
⟨gcpscad⟩ }
⟨gcpscad⟩
```

gcp_endmill_ball

```
⟨gcpscad⟩ module gcp_endmill_ball(es_diameter, es_flute_length) {
⟨gcpscad⟩   translate([0, 0, (es_diameter / 2)]){
⟨gcpscad⟩     union(){
⟨gcpscad⟩       sphere(r=(es_diameter / 2));
⟨gcpscad⟩       cylinder(r1=(es_diameter / 2), r2=(es_diameter / 2), h=es_flute_length, center=fal
⟨gcpscad⟩     }
⟨gcpscad⟩   }
⟨gcpscad⟩ }
⟨gcpscad⟩
```

gcp_endmill_v

```
⟨gcpscad⟩ module gcp_endmill_v(es_v_angle, es_diameter) {
⟨gcpscad⟩   union(){
⟨gcpscad⟩     cylinder(r1=0, r2=(es_diameter / 2), h=((es_diameter / 2) / tan((es_v_angle / 2))),
⟨gcpscad⟩     translate([0, 0, ((es_diameter / 2) / tan((es_v_angle / 2)))]){
⟨gcpscad⟩       cylinder(r1=(es_diameter / 2), r2=(es_diameter / 2), h=((es_diameter * 8) ), cente
⟨gcpscad⟩     }
⟨gcpscad⟩   }
⟨gcpscad⟩ }
⟨gcpscad⟩
```

2.3 File Handling

For writing to files it will be necessary to have commands for each step of working with the files.

popengcodefile
popendxffile
popendxlgblffile
popendxflgsqfile
popendxflgVfile
popendxfsmblfile
popendxfsmsqfile
popendxfsmVfile
popensvgfile

There is a separate function for each type of file, and for DXFs, there are multiple file instances, one for each type of different type and size of tool which it is expected a project will work with.

```
⟨gcpy⟩ def popengcodefile(fn):
⟨gcpy⟩     global f
⟨gcpy⟩     f = open(fn, "w")
⟨gcpy⟩
⟨gcpy⟩ def popendxffile(fn):
⟨gcpy⟩     global dxf
⟨gcpy⟩     dxf = open(fn, "w")
⟨gcpy⟩
⟨gcpy⟩ def popendxlgblffile(fn):
⟨gcpy⟩     global dxflgbl
⟨gcpy⟩     dxflgbl = open(fn, "w")
⟨gcpy⟩
⟨gcpy⟩ def popendxflgsqfile(fn):
⟨gcpy⟩     global dxfldsq
⟨gcpy⟩     dxflgsq = open(fn, "w")
⟨gcpy⟩
⟨gcpy⟩ def popendxflgVfile(fn):
⟨gcpy⟩     global dxflgV
⟨gcpy⟩     dxflgV = open(fn, "w")
⟨gcpy⟩
⟨gcpy⟩ def popendxfsmblfile(fn):
⟨gcpy⟩     global dxfsmbl
⟨gcpy⟩     dxfsmbl = open(fn, "w")
⟨gcpy⟩
⟨gcpy⟩ def popendxfsmsqfile(fn):
⟨gcpy⟩     global dxfsmsq
⟨gcpy⟩     dxfsmsq = open(fn, "w")
⟨gcpy⟩
⟨gcpy⟩ def popendxfsmVfile(fn):
⟨gcpy⟩     global dxfsmV
⟨gcpy⟩     dxfsmV = open(fn, "w")
⟨gcpy⟩
⟨gcpy⟩ def popensvgfile(fn):
⟨gcpy⟩     global svg
⟨gcpy⟩     svg = open(fn, "w")
⟨gcpy⟩
```

oopengcodefile
oopensvgfile
oopendxffile

There will need to be matching OpenSCAD modules for the Python functions.

```
⟨pyscad⟩ module oopengcodefile(fn) {
⟨pyscad⟩  popengcodefile(fn);
⟨pyscad⟩ }
⟨pyscad⟩
⟨pyscad⟩ module oopensvgfile(fn) {
⟨pyscad⟩  popensvgfile(fn);
⟨pyscad⟩ }
⟨pyscad⟩
⟨pyscad⟩ module oopendxffile(fn) {
```

```
414 ⟨pyscad⟩     echo(fn);
415 ⟨pyscad⟩  popendxffile(fn);
416 ⟨pyscad⟩ }
417 ⟨pyscad⟩
418 ⟨pyscad⟩ module oopendxflgblfile(fn) {
419 ⟨pyscad⟩  popendxflgblfile(fn);
420 ⟨pyscad⟩ }
421 ⟨pyscad⟩
422 ⟨pyscad⟩ module oopendxflgsqfile(fn) {
423 ⟨pyscad⟩  popendxflgsqfile(fn);
424 ⟨pyscad⟩ }
425 ⟨pyscad⟩
426 ⟨pyscad⟩ module oopendxflgVfile(fn) {
427 ⟨pyscad⟩  popendxflgVfile(fn);
428 ⟨pyscad⟩ }
429 ⟨pyscad⟩
430 ⟨pyscad⟩ module oopendxfsmblfile(fn) {
431 ⟨pyscad⟩  popendxfsmblfile(fn);
432 ⟨pyscad⟩ }
433 ⟨pyscad⟩
434 ⟨pyscad⟩ module oopendxfsmsqfile(fn) {
435 ⟨pyscad⟩     echo(fn);
436 ⟨pyscad⟩  popendxfsmsqfile(fn);
437 ⟨pyscad⟩ }
438 ⟨pyscad⟩
439 ⟨pyscad⟩ module oopendxfsmVfile(fn) {
440 ⟨pyscad⟩  popendxfsmVfile(fn);
441 ⟨pyscad⟩ }
442 ⟨pyscad⟩
```

opengcodefile
opensvgfile
opendxffile

```
443 ⟨gcpscad⟩ module opengcodefile(fn) {
444 ⟨gcpscad⟩ if (generategcode == true) {
445 ⟨gcpscad⟩  oopengcodefile(fn);
446 ⟨gcpscad⟩     echo(fn);
447 ⟨gcpscad⟩     owritecomment(fn);
448 ⟨gcpscad⟩ }
449 ⟨gcpscad⟩ }
450 ⟨gcpscad⟩
451 ⟨gcpscad⟩ module opensvgfile(fn) {
452 ⟨gcpscad⟩ if (generatesvg == true) {
453 ⟨gcpscad⟩  oopensvgfile(fn);
454 ⟨gcpscad⟩     echo(fn);
455 ⟨gcpscad⟩     svgwriteone(str("<?xml version=",chr(34),"1.0",chr(34)," encoding=",chr(34),"UTF-8",
456 ⟨gcpscad⟩ // writesvglineend();
457 ⟨gcpscad⟩ svgwriteone(str("<svg  version=",chr(34),"1.1",chr(34)," xmlns=",chr(34),"http://www.w3.
458 ⟨gcpscad⟩ //<path d="M755.906 0 L755.906 377.953 L0 377.953 L0 0 L755.906 0 Z " stroke="black" str
459 ⟨gcpscad⟩ svgwriteone(str("<path d=",chr(34),"M",stocklength*3.77953," 0 L",stocklength*3.77953,"
460 ⟨gcpscad⟩     }
461 ⟨gcpscad⟩ }
462 ⟨gcpscad⟩
463 ⟨gcpscad⟩ module opendxffile(fn) {
464 ⟨gcpscad⟩ if (generatedxf == true) {
465 ⟨gcpscad⟩  oopendxffile(str(fn,".dxf"));
466 ⟨gcpscad⟩ //    echo(fn);
```

```
⟨gcpscad⟩     dxfwriteone("0");
⟨gcpscad⟩     dxfwriteone("SECTION");
⟨gcpscad⟩     dxfwriteone("2");
⟨gcpscad⟩     dxfwriteone("ENTITIES");
⟨gcpscad⟩     dxfwriteone("0");
⟨gcpscad⟩ if (large_ball_tool_no >  0) { oopendxflgblfile(str(fn,".",large_ball_tool_no
⟨gcpscad⟩     dxfpreamble(large_ball_tool_no);
⟨gcpscad⟩ }
⟨gcpscad⟩ if (large_square_tool_no >  0) { oopendxflgsqfile(str(fn,".",large_square_tool
⟨gcpscad⟩     dxfpreamble(large_square_tool_no);
⟨gcpscad⟩ }
⟨gcpscad⟩ if (large_V_tool_no >  0) { oopendxflgVfile(str(fn,".",large_V_tool_no,".dxf"
⟨gcpscad⟩     dxfpreamble(large_V_tool_no);
⟨gcpscad⟩ }
⟨gcpscad⟩ if (small_ball_tool_no >  0) { oopendxfsmblfile(str(fn,".",small_ball_tool_no
⟨gcpscad⟩     dxfpreamble(small_ball_tool_no);
⟨gcpscad⟩ }
⟨gcpscad⟩ if (small_square_tool_no >  0) { oopendxfsmsqfile(str(fn,".",small_square_tool
⟨gcpscad⟩ //    echo(str("tool no",small_square_tool_no));
⟨gcpscad⟩     dxfpreamble(small_square_tool_no);
⟨gcpscad⟩ }
⟨gcpscad⟩ if (small_V_tool_no >  0) { oopendxfsmVfile(str(fn,".",small_V_tool_no,".dxf"
⟨gcpscad⟩     dxfpreamble(small_V_tool_no);
⟨gcpscad⟩ }
⟨gcpscad⟩ }
⟨gcpscad⟩ }
⟨gcpscad⟩
```

writedxf
writedxflgbl
writedxflgsq

Once files have been opened they may be written to.

```
⟨gcpy⟩ def writedxf(*arguments):
⟨gcpy⟩     line_to_write = ""
⟨gcpy⟩     for element in arguments:
⟨gcpy⟩         line_to_write += element
⟨gcpy⟩     dxf.write(line_to_write)
⟨gcpy⟩     dxf.write("\n")
⟨gcpy⟩
⟨gcpy⟩ def writedxflgbl(*arguments):
⟨gcpy⟩     line_to_write = ""
⟨gcpy⟩     for element in arguments:
⟨gcpy⟩         line_to_write += element
⟨gcpy⟩     dxflgbl.write(line_to_write)
⟨gcpy⟩     print(line_to_write)
⟨gcpy⟩     dxflgbl.write("\n")
⟨gcpy⟩
⟨gcpy⟩ def writedxflgsq(*arguments):
⟨gcpy⟩     line_to_write = ""
⟨gcpy⟩     for element in arguments:
⟨gcpy⟩         line_to_write += element
⟨gcpy⟩     dxflgsq.write(line_to_write)
⟨gcpy⟩     print(line_to_write)
⟨gcpy⟩     dxflgsq.write("\n")
⟨gcpy⟩
⟨gcpy⟩ def writedxflgV(*arguments):
⟨gcpy⟩     line_to_write = ""
```

```
519 ⟨gcpy⟩     for element in arguments:
520 ⟨gcpy⟩         line_to_write += element
521 ⟨gcpy⟩     dxflgV.write(line_to_write)
522 ⟨gcpy⟩     print(line_to_write)
523 ⟨gcpy⟩     dxflgV.write("\n")
524 ⟨gcpy⟩
525 ⟨gcpy⟩ def writedxfsmbl(*arguments):
526 ⟨gcpy⟩     line_to_write = ""
527 ⟨gcpy⟩     for element in arguments:
528 ⟨gcpy⟩         line_to_write += element
529 ⟨gcpy⟩     dxfsmbl.write(line_to_write)
530 ⟨gcpy⟩     print(line_to_write)
531 ⟨gcpy⟩     dxfsmbl.write("\n")
532 ⟨gcpy⟩
533 ⟨gcpy⟩ def writedxfsmsq(*arguments):
534 ⟨gcpy⟩     line_to_write = ""
535 ⟨gcpy⟩     for element in arguments:
536 ⟨gcpy⟩         line_to_write += element
537 ⟨gcpy⟩     dxfsmsq.write(line_to_write)
538 ⟨gcpy⟩     print(line_to_write)
539 ⟨gcpy⟩     dxfsmsq.write("\n")
540 ⟨gcpy⟩
541 ⟨gcpy⟩ def writedxfsmV(*arguments):
542 ⟨gcpy⟩     line_to_write = ""
543 ⟨gcpy⟩     for element in arguments:
544 ⟨gcpy⟩         line_to_write += element
545 ⟨gcpy⟩     dxfsmV.write(line_to_write)
546 ⟨gcpy⟩     print(line_to_write)
547 ⟨gcpy⟩     dxfsmV.write("\n")
548 ⟨gcpy⟩
549 ⟨gcpy⟩ def writesvg(*arguments):
550 ⟨gcpy⟩     line_to_write = ""
551 ⟨gcpy⟩     for element in arguments:
552 ⟨gcpy⟩         line_to_write += element
553 ⟨gcpy⟩     svg.write(line_to_write)
554 ⟨gcpy⟩     print(line_to_write)
555 ⟨gcpy⟩
556 ⟨gcpy⟩ def pwritesvgline():
557 ⟨gcpy⟩     svg.write("\n")
558 ⟨gcpy⟩
```

owritecomment
dxfwriteone
dxfwritelgbl

```
559 ⟨pyscad⟩ module owritecomment(comment) {
560 ⟨pyscad⟩  writeln("(",comment,")");
561 ⟨pyscad⟩ }
562 ⟨pyscad⟩
563 ⟨pyscad⟩ module dxfwriteone(first) {
564 ⟨pyscad⟩  writedxf(first);
565 ⟨pyscad⟩ // writeln(first);
566 ⟨pyscad⟩ //    echo(first);
567 ⟨pyscad⟩ }
568 ⟨pyscad⟩
569 ⟨pyscad⟩ module dxfwritelgbl(first) {
570 ⟨pyscad⟩  writedxflgbl(first);
```

```
⟨pyscad⟩ }
⟨pyscad⟩
⟨pyscad⟩ module dxfwritelgsq(first) {
⟨pyscad⟩  writedxflgsq(first);
⟨pyscad⟩ }
⟨pyscad⟩
⟨pyscad⟩ module dxfwritelgV(first) {
⟨pyscad⟩  writedxflgV(first);
⟨pyscad⟩ }
⟨pyscad⟩
⟨pyscad⟩ module dxfwritesmbl(first) {
⟨pyscad⟩  writedxfsmbl(first);
⟨pyscad⟩ }
⟨pyscad⟩
⟨pyscad⟩ module dxfwritesmsq(first) {
⟨pyscad⟩  writedxfsmsq(first);
⟨pyscad⟩ }
⟨pyscad⟩
⟨pyscad⟩ module dxfwritesmV(first) {
⟨pyscad⟩  writedxfsmV(first);
⟨pyscad⟩ }
⟨pyscad⟩
⟨pyscad⟩ module svgwriteone(first) {
⟨pyscad⟩  writesvg(first);
⟨pyscad⟩ }
⟨pyscad⟩
⟨pyscad⟩ module writesvglineend(first) {
⟨pyscad⟩  pwritesvgline();
⟨pyscad⟩ }
⟨pyscad⟩
⟨pyscad⟩ module owriteone(first) {
⟨pyscad⟩  writeln(first);
⟨pyscad⟩ }
⟨pyscad⟩
⟨pyscad⟩ module owritetwo(first, second) {
⟨pyscad⟩  writeln(first, second);
⟨pyscad⟩ }
⟨pyscad⟩
⟨pyscad⟩ module owritethree(first, second, third) {
⟨pyscad⟩  writeln(first, second, third);
⟨pyscad⟩ }
⟨pyscad⟩
⟨pyscad⟩ module owritefour(first, second, third, fourth) {
⟨pyscad⟩  writeln(first, second, third, fourth);
⟨pyscad⟩ }
⟨pyscad⟩
⟨pyscad⟩ module owritefive(first, second, third, fourth, fifth) {
⟨pyscad⟩  writeln(first, second, third, fourth, fifth);
⟨pyscad⟩ }
⟨pyscad⟩
⟨pyscad⟩ module owritesix(first, second, third, fourth, fifth, sixth) {
⟨pyscad⟩  writeln(first, second, third, fourth, fifth, sixth);
⟨pyscad⟩ }
⟨pyscad⟩
```

```
625 ⟨pyscad⟩ module owriteseven(first, second, third, fourth, fifth, sixth, seventh) {
626 ⟨pyscad⟩  writeln(first, second, third, fourth, fifth, sixth, seventh);
627 ⟨pyscad⟩ }
628 ⟨pyscad⟩
629 ⟨pyscad⟩ module owriteeight(first, second, third, fourth, fifth, sixth, seventh,eighth) {
630 ⟨pyscad⟩  writeln(first, second, third, fourth, fifth, sixth, seventh,eighth);
631 ⟨pyscad⟩ }
632 ⟨pyscad⟩
633 ⟨pyscad⟩ module owritenine(first, second, third, fourth, fifth, sixth, seventh, eighth, ninth) {
634 ⟨pyscad⟩  writeln(first, second, third, fourth, fifth, sixth, seventh, eighth, ninth);
635 ⟨pyscad⟩ }
636 ⟨pyscad⟩
637 ⟨pyscad⟩ module owriteten(first, second, third, fourth, fifth, sixth, seventh, eighth, ninth, tenth
638 ⟨pyscad⟩  writeln(first, second, third, fourth, fifth, sixth, seventh, eighth, ninth, tenth);
639 ⟨pyscad⟩ }
640 ⟨pyscad⟩
641 ⟨pyscad⟩ module owriteeleven(first, second, third, fourth, fifth, sixth, seventh, eighth, ninth, t
642 ⟨pyscad⟩  writeln(first, second, third, fourth, fifth, sixth, seventh, eighth, ninth, tenth, eleve
643 ⟨pyscad⟩ }
644 ⟨pyscad⟩
645 ⟨pyscad⟩ module owritetwelve(first, second, third, fourth, fifth, sixth, seventh, eighth, ninth, t
646 ⟨pyscad⟩  writeln(first, second, third, fourth, fifth, sixth, seventh, eighth, ninth, tenth, eleve
647 ⟨pyscad⟩ }
648 ⟨pyscad⟩
649 ⟨pyscad⟩ module owritethirteen(first, second, third, fourth, fifth, sixth, seventh, eighth, ninth,
650 ⟨pyscad⟩  writeln(first, second, third, fourth, fifth, sixth, seventh, eighth, ninth, tenth, eleve
651 ⟨pyscad⟩ }
652 ⟨pyscad⟩
```

dxfwrite
dxfpreamble
writesvgline

```
653 ⟨gcpscad⟩ module dxfwrite(tn,arg) {
654 ⟨gcpscad⟩ if (tn == large_ball_tool_no) {
655 ⟨gcpscad⟩     dxfwritelgbl(arg);}
656 ⟨gcpscad⟩ if (tn == large_square_tool_no) {
657 ⟨gcpscad⟩     dxfwritelgsq(arg);}
658 ⟨gcpscad⟩ if (tn == large_V_tool_no) {
659 ⟨gcpscad⟩     dxfwritelgV(arg);}
660 ⟨gcpscad⟩ if (tn == small_ball_tool_no) {
661 ⟨gcpscad⟩     dxfwritesmbl(arg);}
662 ⟨gcpscad⟩ if (tn == small_square_tool_no) {
663 ⟨gcpscad⟩     dxfwritesmsq(arg);}
664 ⟨gcpscad⟩ if (tn == small_V_tool_no) {
665 ⟨gcpscad⟩     dxfwritesmV(arg);}
666 ⟨gcpscad⟩ }
667 ⟨gcpscad⟩
668 ⟨gcpscad⟩ module dxfpreamble(tn) {
669 ⟨gcpscad⟩ //    echo(str("dxfpreamble",small_square_tool_no));
670 ⟨gcpscad⟩     dxfwrite(tn,"0");
671 ⟨gcpscad⟩     dxfwrite(tn,"SECTION");
672 ⟨gcpscad⟩     dxfwrite(tn,"2");
673 ⟨gcpscad⟩     dxfwrite(tn,"ENTITIES");
674 ⟨gcpscad⟩     dxfwrite(tn,"0");
675 ⟨gcpscad⟩ }
676 ⟨gcpscad⟩
677 ⟨gcpscad⟩ module writesvgline(bx,by,ex,ey) {
```

```
678 ⟨gcpscad⟩ if (generatesvg == true) {
679 ⟨gcpscad⟩     svgwriteone(str("<path d=",chr(34),"M",bx*3.77953," ",by*3.77953," L",ex*3
680 ⟨gcpscad⟩     }
681 ⟨gcpscad⟩ }
682 ⟨gcpscad⟩
683 ⟨gcpscad⟩ module dxfbpl(tn,bx,by) {
684 ⟨gcpscad⟩     dxfwrite(tn,"POLYLINE");
685 ⟨gcpscad⟩     dxfwrite(tn,"8");
686 ⟨gcpscad⟩     dxfwrite(tn,"default");
687 ⟨gcpscad⟩     dxfwrite(tn,"66");
688 ⟨gcpscad⟩     dxfwrite(tn,"1");
689 ⟨gcpscad⟩     dxfwrite(tn,"70");
690 ⟨gcpscad⟩     dxfwrite(tn,"0");
691 ⟨gcpscad⟩     dxfwrite(tn,"0");
692 ⟨gcpscad⟩     dxfwrite(tn,"VERTEX");
693 ⟨gcpscad⟩     dxfwrite(tn,"8");
694 ⟨gcpscad⟩     dxfwrite(tn,"default");
695 ⟨gcpscad⟩     dxfwrite(tn,"70");
696 ⟨gcpscad⟩     dxfwrite(tn,"32");
697 ⟨gcpscad⟩     dxfwrite(tn,"10");
698 ⟨gcpscad⟩     dxfwrite(tn,str(bx));
699 ⟨gcpscad⟩     dxfwrite(tn,"20");
700 ⟨gcpscad⟩     dxfwrite(tn,str(by));
701 ⟨gcpscad⟩     dxfwrite(tn,"0");
702 ⟨gcpscad⟩ }
703 ⟨gcpscad⟩
704 ⟨gcpscad⟩ module beginpolyline(bx,by,bz) {
705 ⟨gcpscad⟩ if (generatedxf == true) {
706 ⟨gcpscad⟩     dxfwriteone("POLYLINE");
707 ⟨gcpscad⟩     dxfwriteone("8");
708 ⟨gcpscad⟩     dxfwriteone("default");
709 ⟨gcpscad⟩     dxfwriteone("66");
710 ⟨gcpscad⟩     dxfwriteone("1");
711 ⟨gcpscad⟩     dxfwriteone("70");
712 ⟨gcpscad⟩     dxfwriteone("0");
713 ⟨gcpscad⟩     dxfwriteone("0");
714 ⟨gcpscad⟩     dxfwriteone("VERTEX");
715 ⟨gcpscad⟩     dxfwriteone("8");
716 ⟨gcpscad⟩     dxfwriteone("default");
717 ⟨gcpscad⟩     dxfwriteone("70");
718 ⟨gcpscad⟩     dxfwriteone("32");
719 ⟨gcpscad⟩     dxfwriteone("10");
720 ⟨gcpscad⟩     dxfwriteone(str(bx));
721 ⟨gcpscad⟩     dxfwriteone("20");
722 ⟨gcpscad⟩     dxfwriteone(str(by));
723 ⟨gcpscad⟩     dxfwriteone("0");
724 ⟨gcpscad⟩     dxfbpl(current_tool(),bx,by);}
725 ⟨gcpscad⟩ }
726 ⟨gcpscad⟩
727 ⟨gcpscad⟩ module dxfapl(tn,bx,by) {
728 ⟨gcpscad⟩     dxfwrite(tn,"VERTEX");
729 ⟨gcpscad⟩     dxfwrite(tn,"8");
730 ⟨gcpscad⟩     dxfwrite(tn,"default");
731 ⟨gcpscad⟩     dxfwrite(tn,"70");
```

```
⟨gcpscad⟩     dxfwrite(tn,"32");
⟨gcpscad⟩     dxfwrite(tn,"10");
⟨gcpscad⟩     dxfwrite(tn,str(bx));
⟨gcpscad⟩     dxfwrite(tn,"20");
⟨gcpscad⟩     dxfwrite(tn,str(by));
⟨gcpscad⟩     dxfwrite(tn,"0");
⟨gcpscad⟩ }
⟨gcpscad⟩
⟨gcpscad⟩ module addpolyline(bx,by,bz) {
⟨gcpscad⟩ if (generatedxf == true) {
⟨gcpscad⟩     dxfwriteone("VERTEX");
⟨gcpscad⟩     dxfwriteone("8");
⟨gcpscad⟩     dxfwriteone("default");
⟨gcpscad⟩     dxfwriteone("70");
⟨gcpscad⟩     dxfwriteone("32");
⟨gcpscad⟩     dxfwriteone("10");
⟨gcpscad⟩     dxfwriteone(str(bx));
⟨gcpscad⟩     dxfwriteone("20");
⟨gcpscad⟩     dxfwriteone(str(by));
⟨gcpscad⟩     dxfwriteone("0");
⟨gcpscad⟩     dxfapl(current_tool(),bx,by);
⟨gcpscad⟩     }
⟨gcpscad⟩ }
⟨gcpscad⟩
⟨gcpscad⟩ module dxfcpl(tn) {
⟨gcpscad⟩     dxfwrite(tn,"SEQEND");
⟨gcpscad⟩     dxfwrite(tn,"0");
⟨gcpscad⟩ }
⟨gcpscad⟩
⟨gcpscad⟩ module closepolyline() {
⟨gcpscad⟩ if (generatedxf == true) {
⟨gcpscad⟩     dxfwriteone("SEQEND");
⟨gcpscad⟩     dxfwriteone("0");
⟨gcpscad⟩     dxfcpl(current_tool());
⟨gcpscad⟩     }
⟨gcpscad⟩ }
⟨gcpscad⟩
⟨gcpscad⟩ module writecomment(comment) {
⟨gcpscad⟩ if (generategcode == true) {
⟨gcpscad⟩  owritecomment(comment);
⟨gcpscad⟩ }
⟨gcpscad⟩ }
⟨gcpscad⟩
```

`pclosegcodefile` `pclosesvgfile` `pclosedxffile`

At the end of the project it will be necessary to close each file. In some instances it will be necessary to write additional information, depending on the file format.

```
⟨gcpy⟩ def pclosegcodefile():
⟨gcpy⟩     f.close()
⟨gcpy⟩
⟨gcpy⟩ def pclosesvgfile():
⟨gcpy⟩     svg.close()
⟨gcpy⟩
⟨gcpy⟩ def pclosedxffile():
```

```
⟨gcpy⟩     dxf.close()
⟨gcpy⟩
⟨gcpy⟩ def pclosedxflgblfile():
⟨gcpy⟩     dxflgbl.close()
⟨gcpy⟩
⟨gcpy⟩ def pclosedxflgsqfile():
⟨gcpy⟩     dxflgsq.close()
⟨gcpy⟩
⟨gcpy⟩ def pclosedxflgVfile():
⟨gcpy⟩     dxflgV.close()
⟨gcpy⟩
⟨gcpy⟩ def pclosedxfsmblfile():
⟨gcpy⟩     dxfsmbl.close()
⟨gcpy⟩
⟨gcpy⟩ def pclosedxfsmsqfile():
⟨gcpy⟩     dxfsmsq.close()
⟨gcpy⟩
⟨gcpy⟩ def pclosedxfsmVfile():
⟨gcpy⟩     dxfsmV.close()
⟨gcpy⟩
```

oclosegcodefile
oclosedxffile
oclosedxflgblfile

```
⟨pyscad⟩ module oclosegcodefile() {
⟨pyscad⟩  pclosegcodefile();
⟨pyscad⟩ }
⟨pyscad⟩
⟨pyscad⟩ module oclosedxffile() {
⟨pyscad⟩  pclosedxffile();
⟨pyscad⟩ }
⟨pyscad⟩
⟨pyscad⟩ module oclosedxflgblfile() {
⟨pyscad⟩  pclosedxflgblfile();
⟨pyscad⟩ }
⟨pyscad⟩
⟨pyscad⟩ module oclosedxflgsqfile() {
⟨pyscad⟩  pclosedxflgsqfile();
⟨pyscad⟩ }
⟨pyscad⟩
⟨pyscad⟩ module oclosedxflgVfile() {
⟨pyscad⟩  pclosedxflgVfile();
⟨pyscad⟩ }
⟨pyscad⟩
⟨pyscad⟩ module oclosedxfsmblfile() {
⟨pyscad⟩  pclosedxfsmblfile();
⟨pyscad⟩ }
⟨pyscad⟩
⟨pyscad⟩ module oclosedxfsmsqfile() {
⟨pyscad⟩  pclosedxfsmsqfile();
⟨pyscad⟩ }
⟨pyscad⟩
⟨pyscad⟩ module oclosedxfsmVfile() {
⟨pyscad⟩  pclosedxfsmVfile();
⟨pyscad⟩ }
⟨pyscad⟩
```

```
834 ⟨pyscad⟩ module oclosesvgfile() {
835 ⟨pyscad⟩  pclosesvgfile();
836 ⟨pyscad⟩ }
837 ⟨pyscad⟩
```

closegcodefile
dxfpostamble
closedxffile

```
838 ⟨gcpscad⟩ module closegcodefile() {
839 ⟨gcpscad⟩ if (generategcode == true) {
840 ⟨gcpscad⟩     owriteone("M05");
841 ⟨gcpscad⟩     owriteone("M02");
842 ⟨gcpscad⟩  oclosegcodefile();
843 ⟨gcpscad⟩ }
844 ⟨gcpscad⟩ }
845 ⟨gcpscad⟩
846 ⟨gcpscad⟩ module dxfpostamble(arg) {
847 ⟨gcpscad⟩     dxfwrite(arg,"ENDSEC");
848 ⟨gcpscad⟩     dxfwrite(arg,"0");
849 ⟨gcpscad⟩     dxfwrite(arg,"EOF");
850 ⟨gcpscad⟩ }
851 ⟨gcpscad⟩
852 ⟨gcpscad⟩ module closedxffile() {
853 ⟨gcpscad⟩ if (generatedxf == true) {
854 ⟨gcpscad⟩     dxfwriteone("ENDSEC");
855 ⟨gcpscad⟩     dxfwriteone("0");
856 ⟨gcpscad⟩     dxfwriteone("EOF");
857 ⟨gcpscad⟩  oclosedxffile();
858 ⟨gcpscad⟩     echo("CLOSING");
859 ⟨gcpscad⟩ if (large_ball_tool_no >  0) { dxfpostamble(large_ball_tool_no);
860 ⟨gcpscad⟩     oclosedxflgblfile();
861 ⟨gcpscad⟩ }
862 ⟨gcpscad⟩ if (large_square_tool_no >  0) { dxfpostamble(large_square_tool_no);
863 ⟨gcpscad⟩     oclosedxflgsqfile();
864 ⟨gcpscad⟩ }
865 ⟨gcpscad⟩ if (large_V_tool_no >  0) { dxfpostamble(large_V_tool_no);
866 ⟨gcpscad⟩     oclosedxflgVfile();
867 ⟨gcpscad⟩ }
868 ⟨gcpscad⟩ if (small_ball_tool_no >  0) { dxfpostamble(small_ball_tool_no);
869 ⟨gcpscad⟩     oclosedxfsmblfile();
870 ⟨gcpscad⟩ }
871 ⟨gcpscad⟩ if (small_square_tool_no >  0) { dxfpostamble(small_square_tool_no);
872 ⟨gcpscad⟩     oclosedxfsmsqfile();
873 ⟨gcpscad⟩ }
874 ⟨gcpscad⟩ if (small_V_tool_no >  0) { dxfpostamble(small_V_tool_no);
875 ⟨gcpscad⟩     oclosedxfsmVfile();
876 ⟨gcpscad⟩ }
877 ⟨gcpscad⟩     }
878 ⟨gcpscad⟩ }
879 ⟨gcpscad⟩
880 ⟨gcpscad⟩ module closesvgfile() {
881 ⟨gcpscad⟩ if (generatesvg == true) {
882 ⟨gcpscad⟩     svgwriteone("</svg> ");
883 ⟨gcpscad⟩  oclosesvgfile();
884 ⟨gcpscad⟩     echo("CLOSING SVG");
885 ⟨gcpscad⟩     }
886 ⟨gcpscad⟩ }
```

887 ⟨gcpscad⟩

2.4 Movement and Cutting

otm
ocut
orapid

With all the scaffolding in place, it is possible to model tool movement and cutting and to write out files which represent the desired machine motions.

```
888 ⟨gcpscad⟩ module otm(ex, ey, ez, r,g,b) {
889 ⟨gcpscad⟩ color([r,g,b]) hull(){
890 ⟨gcpscad⟩     translate([xpos(), ypos(), zpos()]){
891 ⟨gcpscad⟩       select_tool(current_tool());
892 ⟨gcpscad⟩     }
893 ⟨gcpscad⟩     translate([ex, ey, ez]){
894 ⟨gcpscad⟩       select_tool(current_tool());
895 ⟨gcpscad⟩     }
896 ⟨gcpscad⟩   }
897 ⟨gcpscad⟩ oset(ex, ey, ez);
898 ⟨gcpscad⟩ }
899 ⟨gcpscad⟩
900 ⟨gcpscad⟩ module ocut(ex, ey, ez) {
901 ⟨gcpscad⟩ //color([0.2,1,0.2]) hull(){
902 ⟨gcpscad⟩ otm(ex, ey, ez, 0.2,1,0.2);
903 ⟨gcpscad⟩ }
904 ⟨gcpscad⟩
905 ⟨gcpscad⟩ module orapid(ex, ey, ez) {
906 ⟨gcpscad⟩ //color([0.93,0,0]) hull(){
907 ⟨gcpscad⟩ otm(ex, ey, ez, 0.93,0,0);
908 ⟨gcpscad⟩ }
909 ⟨gcpscad⟩
910 ⟨gcpscad⟩ module rapidbx(bx, by, bz, ex, ey, ez) {
911 ⟨gcpscad⟩ // writeln("G0 X",bx," Y", by, "Z", bz);
912 ⟨gcpscad⟩ if (generategcode == true) {
913 ⟨gcpscad⟩  writecomment("rapid");
914 ⟨gcpscad⟩  owritesix("G0 X",str(ex)," Y", str(ey), " Z", str(ez));
915 ⟨gcpscad⟩ }
916 ⟨gcpscad⟩     orapid(ex, ey, ez);
917 ⟨gcpscad⟩ }
918 ⟨gcpscad⟩
919 ⟨gcpscad⟩ module rapid(ex, ey, ez) {
920 ⟨gcpscad⟩ // writeln("G0 X",bx," Y", by, "Z", bz);
921 ⟨gcpscad⟩ if (generategcode == true) {
922 ⟨gcpscad⟩  writecomment("rapid");
923 ⟨gcpscad⟩  owritesix("G0 X",str(ex)," Y", str(ey), " Z", str(ez));
924 ⟨gcpscad⟩ }
925 ⟨gcpscad⟩     orapid(ex, ey, ez);
926 ⟨gcpscad⟩ }
927 ⟨gcpscad⟩
928 ⟨gcpscad⟩ module movetosafez() {
929 ⟨gcpscad⟩ //this should be move to retract height
930 ⟨gcpscad⟩ if (generategcode == true) {
931 ⟨gcpscad⟩  writecomment("Move to safe Z to avoid workholding");
932 ⟨gcpscad⟩     owriteone("G53G0Z-5.000");
933 ⟨gcpscad⟩ }
934 ⟨gcpscad⟩     orapid(getxpos(), getypos(), retractheight+55);
```

```
⟨gcpscad⟩ }
⟨gcpscad⟩
⟨gcpscad⟩ module begintoolpath(bx,by,bz) {
⟨gcpscad⟩ if (generategcode == true) {
⟨gcpscad⟩  writecomment("PREPOSITION FOR RAPID PLUNGE");
⟨gcpscad⟩     owritefour("G0X", str(bx), "Y",str(by));
⟨gcpscad⟩     owritetwo("Z", str(bz));
⟨gcpscad⟩     }
⟨gcpscad⟩     orapid(bx,by,bz);
⟨gcpscad⟩ }
⟨gcpscad⟩
⟨gcpscad⟩ module movetosafeheight() {
⟨gcpscad⟩ //this should be move to machine position
⟨gcpscad⟩ if (generategcode == true) {
⟨gcpscad⟩ // writecomment("PREPOSITION FOR RAPID PLUNGE");Z25.650
⟨gcpscad⟩ //G1Z24.663F381.0 ,"F",str(plunge)
⟨gcpscad⟩ if (zeroheight == "Top") {
⟨gcpscad⟩     owritetwo("Z",str(retractheight));
⟨gcpscad⟩ }
⟨gcpscad⟩ }
⟨gcpscad⟩     orapid(getxpos(), getypos(), retractheight+55);
⟨gcpscad⟩ }
⟨gcpscad⟩
⟨gcpscad⟩ module cutoneaxis_setfeed(axis,depth,feed) {
⟨gcpscad⟩ if (generategcode == true) {
⟨gcpscad⟩ // writecomment("PREPOSITION FOR RAPID PLUNGE");Z25.650
⟨gcpscad⟩ //G1Z24.663F381.0 ,"F",str(plunge) G1Z7.612F381.0
⟨gcpscad⟩ if (zeroheight == "Top") {
⟨gcpscad⟩     owritefive("G1",axis,str(depth),"F",str(feed));
⟨gcpscad⟩ }
⟨gcpscad⟩ }
⟨gcpscad⟩ if (axis == "X") {setxpos(depth);}
⟨gcpscad⟩ if (axis == "Y") {setypos(depth);}
⟨gcpscad⟩ if (axis == "Z") {setzpos(depth);}
⟨gcpscad⟩ }
⟨gcpscad⟩
⟨gcpscad⟩ module cut(ex, ey, ez) {
⟨gcpscad⟩ // writeln("G0 X",bx," Y", by, "Z", bz);
⟨gcpscad⟩ if (generategcode == true) {
⟨gcpscad⟩ // writecomment("rapid");
⟨gcpscad⟩  owritesix("G1 X",str(ex)," Y", str(ey), " Z", str(ez));
⟨gcpscad⟩ }
⟨gcpscad⟩ if (generatesvg == true) {
⟨gcpscad⟩ // owritesix("G1 X",str(ex)," Y", str(ey), " Z", str(ez));
⟨gcpscad⟩ //    orapid(getxpos(), getypos(), retractheight+5);
⟨gcpscad⟩     writesvgline(getxpos(),getypos(),ex,ey);
⟨gcpscad⟩ }
⟨gcpscad⟩ ocut(ex, ey, ez);
⟨gcpscad⟩ }
⟨gcpscad⟩
⟨gcpscad⟩ module cutwithfeed(ex, ey, ez, feed) {
⟨gcpscad⟩ // writeln("G0 X",bx," Y", by, "Z", bz);
⟨gcpscad⟩ if (generategcode == true) {
⟨gcpscad⟩ // writecomment("rapid");
```

```
989 ⟨gcpscad⟩  owriteeight("G1 X",str(ex)," Y", str(ey), " Z", str(ez),"F",str(feed));
990 ⟨gcpscad⟩ }
991 ⟨gcpscad⟩ ocut(ex, ey, ez);
992 ⟨gcpscad⟩ }
993 ⟨gcpscad⟩
994 ⟨gcpscad⟩ module endtoolpath() {
995 ⟨gcpscad⟩ if (generategcode == true) {
996 ⟨gcpscad⟩ //Z31.750
997 ⟨gcpscad⟩ // owriteone("G53G0Z-5.000");
998 ⟨gcpscad⟩     owritetwo("Z",str(retractheight));
999 ⟨gcpscad⟩ }
1000 ⟨gcpscad⟩     orapid(getxpos(),getypos(),retractheight);
1001 ⟨gcpscad⟩ }
```

3 gcodepreview_template.scad

```
1002 ⟨gcptmpl⟩ //!OpenSCAD
1003 ⟨gcptmpl⟩
1004 ⟨gcptmpl⟩ use <gcodepreview.py>;
1005 ⟨gcptmpl⟩ use <pygcodepreview.scad>;
1006 ⟨gcptmpl⟩ include <gcodepreview.scad>;
1007 ⟨gcptmpl⟩
1008 ⟨gcptmpl⟩ $fa = 2;
1009 ⟨gcptmpl⟩ $fs = 0.125;
1010 ⟨gcptmpl⟩
1011 ⟨gcptmpl⟩ /* [Export] */
1012 ⟨gcptmpl⟩ Base_filename = "export";
1013 ⟨gcptmpl⟩
1014 ⟨gcptmpl⟩ /* [Export] */
1015 ⟨gcptmpl⟩ generatedxf = true;
1016 ⟨gcptmpl⟩
1017 ⟨gcptmpl⟩ /* [Export] */
1018 ⟨gcptmpl⟩ generategcode = true;
1019 ⟨gcptmpl⟩
1020 ⟨gcptmpl⟩ /* [Export] */
1021 ⟨gcptmpl⟩ generatesvg = false;
1022 ⟨gcptmpl⟩
1023 ⟨gcptmpl⟩ /* [CAM] */
1024 ⟨gcptmpl⟩ toolradius = 1.5875;
1025 ⟨gcptmpl⟩ /* [CAM] */
1026 ⟨gcptmpl⟩ large_ball_tool_no = 0; // [0:0,111:111,101:101,202:202]
1027 ⟨gcptmpl⟩
1028 ⟨gcptmpl⟩ /* [CAM] */
1029 ⟨gcptmpl⟩ large_square_tool_no = 0; // [0:0,112:112,102:102,201:201]
1030 ⟨gcptmpl⟩
1031 ⟨gcptmpl⟩ /* [CAM] */
1032 ⟨gcptmpl⟩ large_V_tool_no = 0; // [0:0,301:301,690:690]
1033 ⟨gcptmpl⟩
1034 ⟨gcptmpl⟩ /* [CAM] */
1035 ⟨gcptmpl⟩ small_ball_tool_no = 0; // [0:0,121:121,111:111,101:101]
1036 ⟨gcptmpl⟩
1037 ⟨gcptmpl⟩ /* [CAM] */
1038 ⟨gcptmpl⟩ small_square_tool_no = 102; // [0:0,122:122,112:112,102:102]
```

```
⟨gcptmpl⟩
⟨gcptmpl⟩ /* [CAM] */
⟨gcptmpl⟩ small_V_tool_no = 0; // [0:0,390:390,301:301]
⟨gcptmpl⟩
⟨gcptmpl⟩ /* [Feeds and Speeds] */
⟨gcptmpl⟩ plunge = 100;
⟨gcptmpl⟩ /* [Feeds and Speeds] */
⟨gcptmpl⟩ feed = 400;
⟨gcptmpl⟩ /* [Feeds and Speeds] */
⟨gcptmpl⟩ speed = 16000;
⟨gcptmpl⟩ /* [Feeds and Speeds] */
⟨gcptmpl⟩ square_ratio = 1.0; // [0.25:2]
⟨gcptmpl⟩ /* [Feeds and Speeds] */
⟨gcptmpl⟩ small_V_ratio = 0.75; // [0.25:2]
⟨gcptmpl⟩ /* [Feeds and Speeds] */
⟨gcptmpl⟩ large_V_ratio = 0.875; // [0.25:2]
⟨gcptmpl⟩
⟨gcptmpl⟩ /* [Stock] */
⟨gcptmpl⟩ stocklength = 219;
⟨gcptmpl⟩ /* [Stock] */
⟨gcptmpl⟩ stockwidth = 150;
⟨gcptmpl⟩ /* [Stock] */
⟨gcptmpl⟩ stockthickness = 8.35;
⟨gcptmpl⟩ /* [Stock] */
⟨gcptmpl⟩ zeroheight = "Top"; // [Top, Bottom]
⟨gcptmpl⟩ /* [Stock] */
⟨gcptmpl⟩ stockorigin = "Center"; // [Lower-Left, Center-Left, Top-Left, Center]
⟨gcptmpl⟩ /* [Stock] */
⟨gcptmpl⟩ retractheight = 9;
⟨gcptmpl⟩
⟨gcptmpl⟩ filename_gcode = str(Base_filename, ".nc");
⟨gcptmpl⟩ filename_dxf = str(Base_filename);
⟨gcptmpl⟩ filename_svg = str(Base_filename, ".svg");
⟨gcptmpl⟩
⟨gcptmpl⟩ opengcodefile(filename_gcode);
⟨gcptmpl⟩ opendxffile(filename_dxf);
⟨gcptmpl⟩
⟨gcptmpl⟩ difference() {
⟨gcptmpl⟩ setupstock(stocklength, stockwidth, stockthickness, zeroheight, stockorigin);
⟨gcptmpl⟩
⟨gcptmpl⟩ movetosafez();
⟨gcptmpl⟩
⟨gcptmpl⟩ toolchange(small_square_tool_no,speed * square_ratio);
⟨gcptmpl⟩
⟨gcptmpl⟩ begintoolpath(0,0,0.25);
⟨gcptmpl⟩ beginpolyline(0,0,0.25);
⟨gcptmpl⟩
⟨gcptmpl⟩ cutoneaxis_setfeed("Z",-1,plunge*square_ratio);
⟨gcptmpl⟩
⟨gcptmpl⟩ cutwithfeed(stocklength/2,stockwidth/2,-stockthickness,feed);
⟨gcptmpl⟩ addpolyline(stocklength/2,stockwidth/2,-stockthickness);
⟨gcptmpl⟩
⟨gcptmpl⟩ endtoolpath();
⟨gcptmpl⟩ closepolyline();
```

```
1093 ⟨gcptmpl⟩ }
1094 ⟨gcptmpl⟩
1095 ⟨gcptmpl⟩ closegcodefile();
1096 ⟨gcptmpl⟩ closedxffile();
```

References

[RS274] Thomas R. Kramer, Frederick M. Proctor, Elena R. Messina. https://tsapps.nist.gov/publication/get_pdf.cfm?pub_id=823374

Index

Numbers written in italic refer to the page where the corresponding entry is described; numbers underlined refer to the code line of the definition; numbers in roman refer to the code lines where the entry is used.