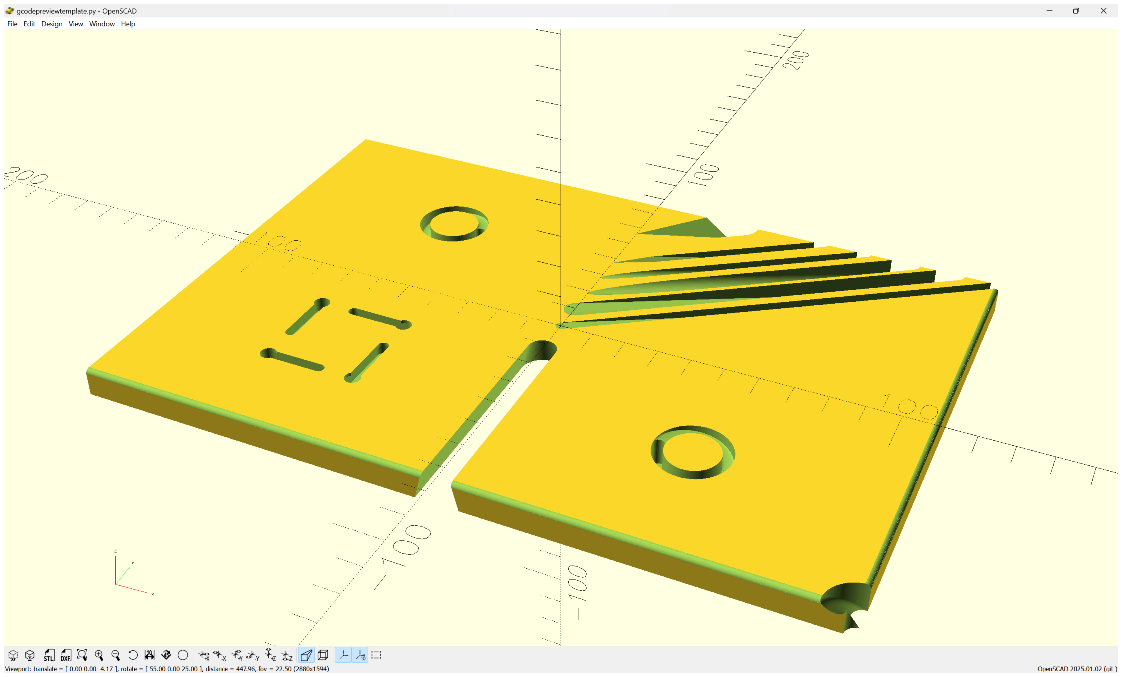
gcodepreview

PythonSCAD library for moving a tool in lines and arcs so as to model how a part would be cut using G-Code, so as to allow PythonSCAD to function as a compleat CAD/CAM solution for subtractive 3-axis CNC (mills and routers) by writing out G-code in addition to 3D modeling (in some cases toolpaths which would not normally be feasible), and to write out DXF files which may be imported into a traditional CAM program to create toolpaths.


gcodepreviewtemplate.py - OpenSCAD
File Edit Design View Window Help
200
100
-100
Viewport: translate = [0.00 0.00 -4.17], rotate = [55.00 0.00 25.00], distance = 447.96, fov = 22.50 (2880x1594)
OpenSCAD 2025.01.02 (git)

Updated to make use of Python in OpenSCAD:[1]

https://pythonscad.org/ (previously this was http://www.guenther-sohler.net/openscad/)

A BlockSCAD file for the initial version of the main modules is available at:

https://www.blockscad3d.com/community/projects/1244473

The project is discussed at:

https://willadams.gitbook.io/design-into-3d/programming

Since it is now programmed using Literate Programming (initially a .dtx, now a .tex file) there is a PDF: https://github.com/WillAdams/gcodepreview/blob/main/gcodepreview.pdf which includes all of the source code with formatted commentary.

The files for this library are:

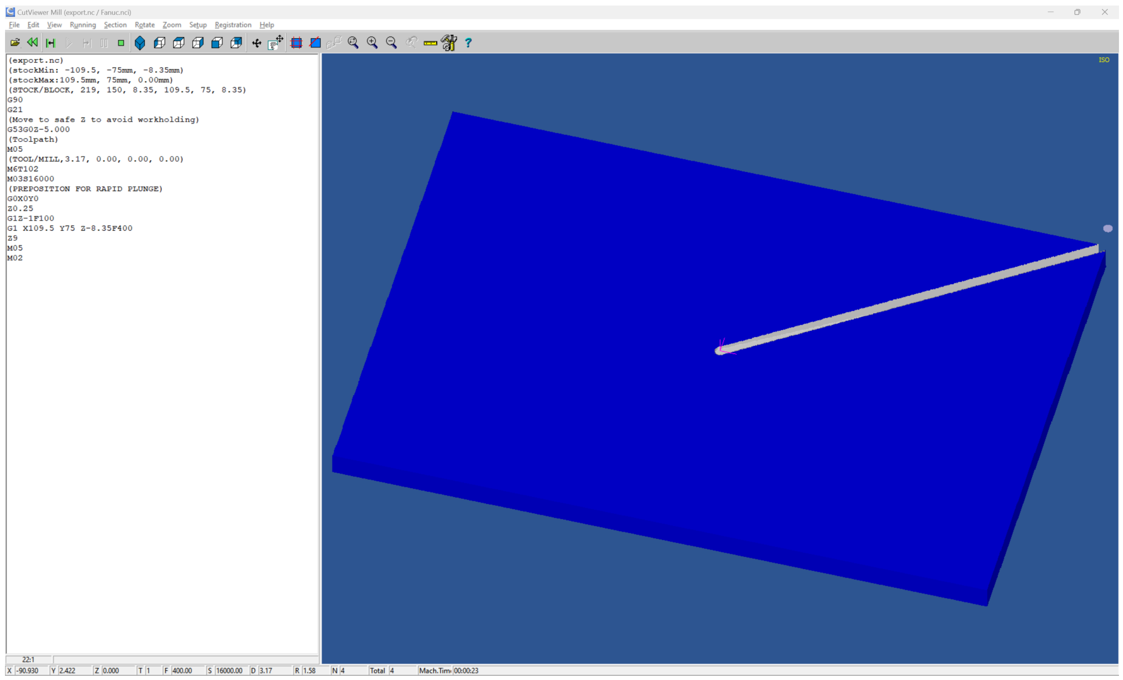
- gcodepreview.py (gcpy) --- the Python functions and variables
- gcodepreview.scad (gcpscad) --- OpenSCAD modules and variables
- gcodepreviewtemplate.scad (gcptmpl) --- .scad example file
- gcodepreviewtemplate.py (gcptmplpy) --- .py example file (which requires PythonSCAD)
- gcpdxf.py (gcpdxfpy) --- .py example file which only makes dxf file(s) and which will run in "normal" Python

If using from PythonSCAD, place the files in C:\Users\~\Documents\OpenSCAD\libraries and call as:[2]

Note that it is necessary to use the first file (this allows loading the Python commands (it used to be necessary to use an intermediary .scad file so as to wrap them in OpenSCAD commands) and then include the last file (which allows using OpenSCAD variables to selectively implement the Python commands via their being wrapped in OpenSCAD modules) and define variables which match the project and then use commands such as:

```
opengcodefile(Gcode_filename);
opendxffile(DXF_filename);

gcp = gcodepreview(true, true, true);

setupstock(219, 150, 8.35, "Top", "Center");

movetosafeZ();

toolchange(102,17000);

cutline(219/2,150/2,-8.35);

stockandtoolpaths();

closegcodefile();
closedxffile();
```

which makes a G-code file:


CutViewer Mill (export.nc / Fanuc.nci)
File Edit View Running Section Rotate Zoom Setup Registration Help
ISO

```
(export.nc)
(stockMin: -109.5, -75mm, -8.35mm)
(stockMax:109.5mm, 75mm, 0.00mm)
(STOCK/BLOCK, 219, 150, 8.35, 109.5, 75, 8.35)
G90
G21
(Move to safe Z to avoid workholding)
G53G0Z-5.000
(Toolpath)
M05
(TOOL/MILL,3.17, 0.00, 0.00, 0.00)
M6T102
M03S16000
(PREPOSITION FOR RAPID PLUNGE)
G0X0Y0
Z0.25
G1Z-1F100
G1 X109.5 Y75 Z-8.35F400
Z9
M05
M02
```

```
22:1
X -90.930  Y 2.422  Z 0.000  T 1  F 400.00  S 16000.00  D 3.17  R 1.58  N 4  Total 4  Mach.Tim 00:00:23
```

but one which could only be sent to a machine so as to cut only the softest and most yielding of materials since it makes a single full-depth pass, and of which has a matching DXF which may be imported into a CAM tool --- but which it is not directly possible to assign a toolpath in readily available CAM tools (since it varies in depth from beginning-to-end).

Importing this DXF and actually cutting it is discussed at:

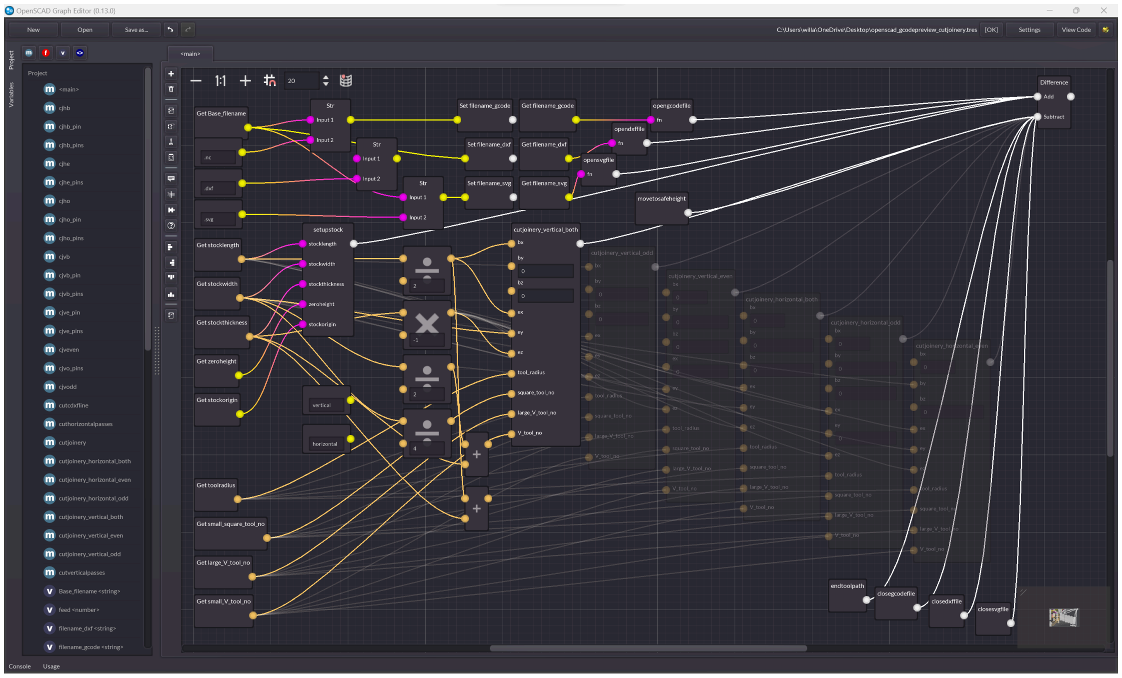
https://forum.makerforums.info/t/rewriting-gcodepreview-with-python/88617/14

Alternately, gcodepreview.py may be placed in a Python library location and used directly from Python --- note that it is possible to use it from a "normal" Python when generating only DXFs.

Tool numbers match those of tooling sold by Carbide 3D (ob. discl., I work for them).

Comments are included in the G-code to match those expected by CutViewer, allowing a direct preview without the need to maintain a tool library.

Supporting OpenSCAD usage makes possible such examples as: openscad_gcodepreview_cutjoinery.tres.scad which is made from an OpenSCAD Graph Editor file:



Version	Notes
0.1	Version supports setting up stock, origin, rapid positioning, making cuts, and writing out matching G-code, and creating a DXF with polylines.
	- separate dxf files are written out for each tool where tool is ball/square/V and small/large (10/31/23)
	- re-writing as a Literate Program using the LaTeX package docmfp (begun 4/12/24)
	- support for additional tooling shapes such as dovetail and keyhole tools
0.2	Adds support for arcs, specialty toolpaths such as Keyhole which may be used for dovetail as well as keyhole cutters
0.3	Support for curves along the 3rd dimension, roundover tooling
0.4	Rewrite using literati documentclass, suppression of SVG code, dxfrectangle
0.5	More shapes, consolidate rectangles, arcs, and circles in gcodepreview.scad
0.6	Notes on modules, change file for setupstock
0.61	Validate all code so that it runs without errors from sample (NEW: Note that this version is archived as gcodepreview-openscad_0_6.tex and the matching PDF is available as well
0.7	Re-write completely in Python
0.8	Re-re-write completely in Python and OpenSCAD, iteratively testing

Possible future improvements:

- support for additional tooling shapes (bowl bits with flat bottom, tapered ball nose, lollipop cutters)
- create a single line font for use where text is wanted
- Support Bézier curves (required for fonts if not to be limited to lines and arcs) and surfaces

Note for G-code generation that it is up to the user to implement Depth per Pass so as to not take a single full-depth pass as noted above. Working from a DXF of course allows one to off-load such considerations to a specialized CAM tool.

Deprecated feature:

- exporting SVGs --- coordinate system differences between OpenSCAD/DXFs and SVGs would require managing the inversion of the coordinate system (using METAPOST, which shares the same orientation and which can write out SVGs may be used for future versions)

To-do:

- fix line numbers
- fix OpenSCAD wrapper
- reposition cutroundover command into cutshape
- work on rotary axis option

1. Previous versions had used RapCAD, so as to take advantage of the writeln command, which has since been re-written in Python. ↩

2. C:\Users\~\Documents\RapCAD\libraries is deprecated since RapCAD is no longer needed since Python is now used for writing out files)

   use <gcodepreview.py> include <gcodepreview.scad> ↩