WillAdams /
gcodepreview

Code Issues Pull requests Actions Projects Wiki Security Insights Settings

OpenSCAD library for moving a tool in lines and arcs so as to model how a part would be cut using G-Code or described as a DXF.

LGPL-2.1 license

14 stars 3 forks 3 watching 1 Branch 0 Tags Activity

Public repository

main 1 Branch 0 Tags Go to file t Go to file Add file Code

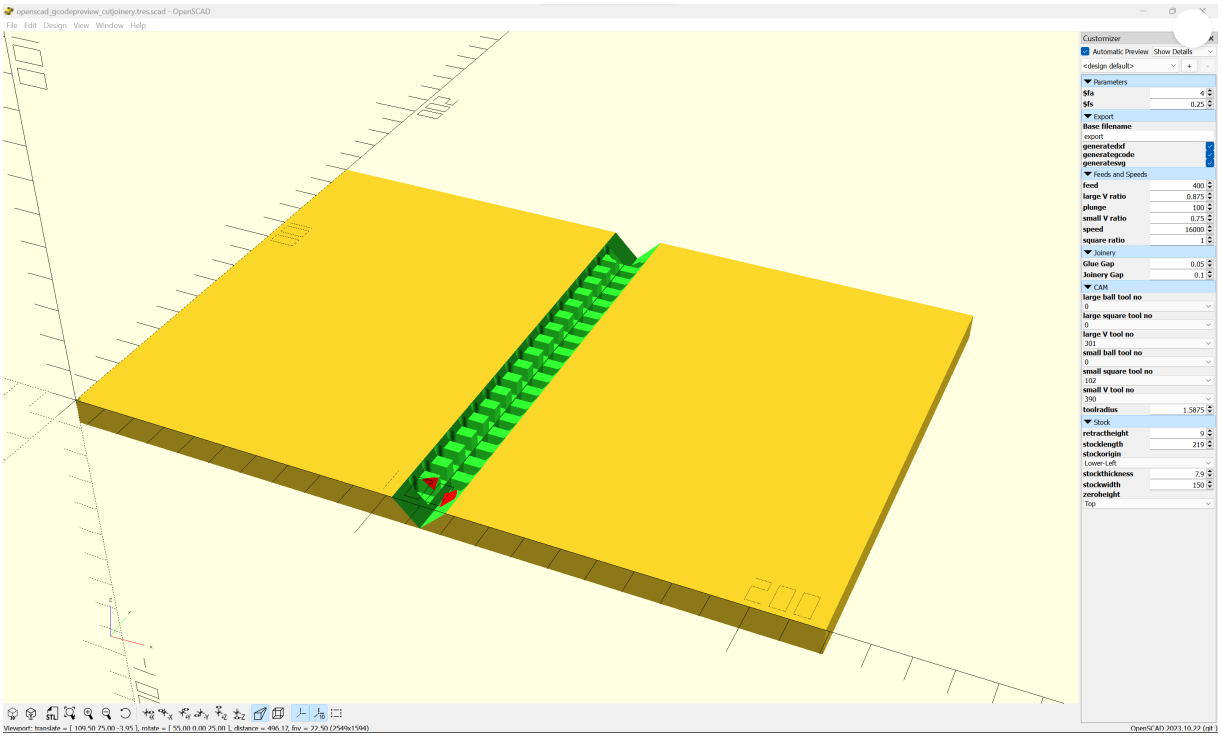
WillAdams Delete platexmkrc 5dc9c0e · now

__pycache__	Literate corrections	3 months ago
LICENSE	Initial commit	3 years ago
OSGE_cutjoinery.png	Small business Saturday	8 months ago
README.md	Independence Day colons	2 weeks ago
cut2Dshapes.scad	Flag Day	last month
cut2Dshapes.tres	Small business Saturday	8 months ago
export.102.dxf	Literate corrections	3 months ago
export.dxf	Literate corrections	3 months ago
export.nc	Literate corrections	3 months ago
flatten.graph.tres	Add files via upload	last year
gcode_flatten.rcad	setupstock, not setupcut	2 years ago
gcodepreview.drv	Literate corrections	3 months ago
gcodepreview.dtx	Matariki	3 weeks ago
gcodepreview.ins	Flag Day	last month
gcodepreview.log	Literate corrections	3 months ago
gcodepreview.pdf	needs index generation	2 weeks ago
gcodepreview.py	Matariki	3 weeks ago
gcodepreview.scad	Matariki	3 weeks ago
gcodepreview_template.png	Create gcodepreview_template.png	3 months ago
gcodepreview_template.scad	Flag Day	last month
gcodepreviewing.scad	Update gcodepreviewing.scad	2 years ago
gcoderadiuspreview.scad	Add files via upload	3 years ago
gcp_template.graph.tres	Add files via upload	6 months ago
openscad_cutjoinery.png	Small business Saturday	8 months ago
openscad_gcodepreview.scad	Update and rename gcodepreview.scad to openscad_gco...	10 months ago
openscad_gcodepreview_cutjoinery.tres	snow day fix	6 months ago
openscad_gcodepreview_cutjoinery.tres.scad	snow day fix	6 months ago
osge_cutjoinery.png	Small business Saturday	8 months ago
pygcodepreview.scad	Flag Day	last month

README LGPL-2.1 license

gcodepreview

OpenSCAD 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 OpenSCAD to function as a compleat CAD/CAM solution for subtractive CNC (mills and routers) by writing out G-code (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.



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://forum.makerforums.info/t/g-code-preview-using-openscad-rapcad/85729

and

https://forum.makerforums.info/t/openscad-and-python-looking-to-finally-be-resolved/88171

and

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

Since it is now programmed using Literate Programming (.dtx) 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:

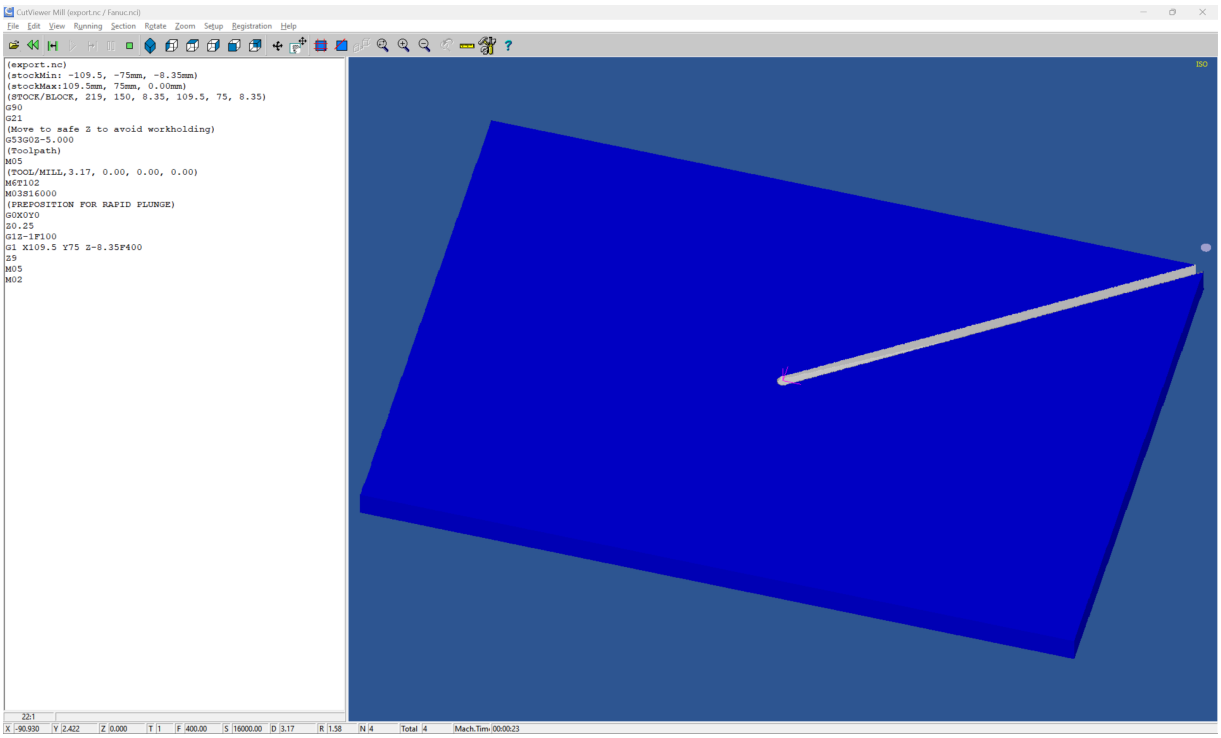
- gcodepreview.py (gcpy) --- the Python core
- pygcodepreview.scad (pyscad) --- the Python functions wrapped in OpenSCAD
- gcodepreview.scad(gcpscad) --- OpenSCAD modules and variables
- gcodepreview_template.scad (gcptmpl) --- example file
- cut2Dshapes.scad (cut2D) --- code for cutting 2D shapes

Place the files in C:\Users\~\Documents\OpenSCAD\libraries and call as:[2]

Note that it is necessary to use the first two files (this allows loading the Python commands and then wrapping 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);

difference() {
    setupstock(stocklength, stockwidth, stockthickness, zeroheight, stockorigin);

movetosafez();

toolchange(squaretoolno,speed * square_ratio);

begintoolpath(0,0,0.25);
beginpolyline(0,0,0.25);

cutoneaxis_setfeed("Z",-1,plunge*square_ratio);
addpolyline(stocklength/2,stockwidth/2,-stockthickness);

cutwithfeed(stocklength/2,stockwidth/2,-stockthickness,feed);

endtoolpath();
endpolyline();

}

closegcodefile();
closedxffile();
```

which makes a G-code file:



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).

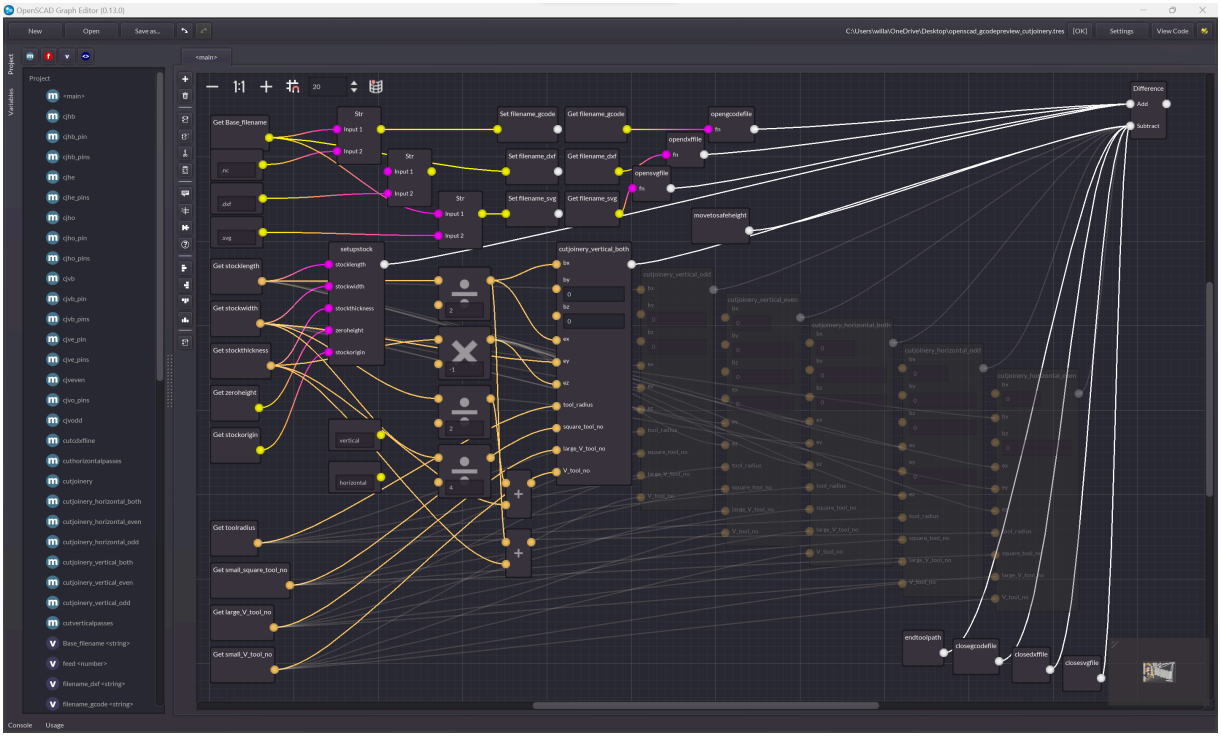
Importing this DXF and actually cutting it is discussed at:

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

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.

A complete example file is: gcodepreview_template.scad and another example is openscad_gcodepreview_cutjoinery.tres.scad which is made from an OpenSCAD Graph Editor file:



Version 0.1 supports setting up stock, origin, rapid positioning, making cuts, and writing out matching G-code, and creating a DXF with polylines.

Added features since initial upload:

- endpolyline(); --- this command allows ending one polyline so as to allow multiple lines in a DXF
- 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

Version 0.2 adds support for arcs

- DXF: support for arcs (which may be used to make circles) (6/1/24)
- Specialty toolpaths such as Keyhole which may be used for dovetail as well as keyhole cutters
- Support for curves along the 3rd dimension
- support for roundover tooling

Possible future improvements:

- support for additional tooling shapes such as tapered ball-nose tools or lollipop cutters or thread-cutting tools
- G-code: support for G2/G3 arcs and circles
- G-code: import external tool libraries and feeds and speeds from JSON or CSV files ---
- general coding improvements --- current coding style is quite prosaic
- generalized modules for cutting out various shapes/geometries --- an in-process one is to cut a rectangular area as vertical passes (the horizontal version will be developed presently)

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. Working from a DXF of course allows one to off-load such considerations to a specialized CAM tool.

Deprecated feature:

- exporting SVGs --- while this was begun, it turns out that these are written out upside down due to coordinate system differences between OpenSCAD/DXFs and SVGs (it is possible that METAPOST will be used instead for future versions)

---

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

Languages

OpenSCAD 50.0% TeX 47.4% Python 2.4% nesC 0.2%

Suggested workflows

Based on your tech stack

Django Configure

Build and Test a Django Project

Python Package using Anaconda Configure

Create and test a Python package on multiple Python versions using Anaconda for package management.

SLSA Generic generator Configure

Generate SLSA3 provenance for your existing release workflows

More workflows

Dismiss suggestions