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

15 stars **3** forks **3** watching **1** Branch **0** Tags Activity

Public repository

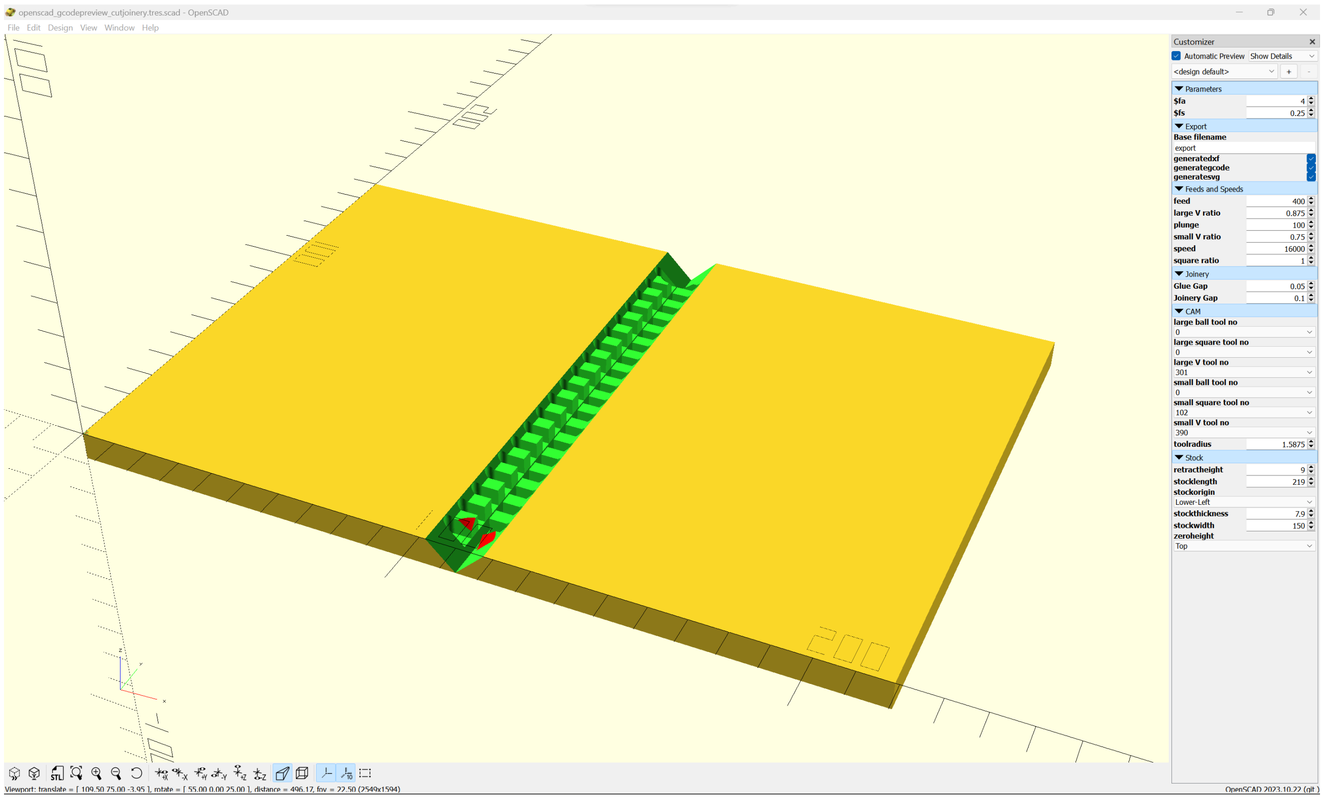
WillAdams National Virginia Day after f879250 · 1 minute ago

images	National Virginia Day	yesterday
.gitignore	Update .gitignore	3 weeks ago
LICENSE	Initial commit	3 years ago
OSGE_cutjoinery.png	Small business Saturday	10 months ago
README.md	Folklore Day	3 weeks ago
cut2Dshapes.scad	Folklore Day	3 weeks ago
cut2Dshapes.tres	Small business Saturday	10 months ago
export.102.dxf	Literate corrections	5 months ago
export.dxf	Literate corrections	5 months ago
export.nc	Literate corrections	5 months ago
flatten.graph.tres	Add files via upload	last year
gcodepreview.pdf	National Virginia Day after	1 minute ago
gcodepreview.py	National Virginia Day	yesterday
gcodepreview.scad	National Virginia Day	yesterday
gcodepreview.tex	National Virginia Day after	1 minute ago
gcodepreview_PYunittests.scad	National Virginia Day after	1 minute ago
gcodepreview_template.png	Create gcodepreview_template.png	5 months ago
gcodepreview_unittests.png	National Virginia Day	2 hours ago
gcodepreviewtemplate.scad	National Virginia Day after	1 minute ago
gcp_template.graph.tres	Add files via upload	8 months ago
literati.sty	National Ampersand Day	last week
openscad_cutjoinery.png	Small business Saturday	10 months ago
openscad_gcodepreview_cutjoinery.tres	snow day fix	8 months ago
osge_cutjoinery.png	Small business Saturday	10 months ago
pygcodepreview.scad	National Virginia Day	yesterday
readme.pdf	International Book Lover's Day PDFs	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 3-axis 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.


openscad_gcodepreview_cutjoinery.tres.scad - OpenSCAD
File Edit Design View Window Help
Customizer
Automatic Preview
Show Details
<design default>
Viewport: translate = [109.50 75.00 -3.95], rotate = [55.00 0.00 25.00], distance = 496.17, fov = 22.50 (2549x1594)
OpenSCAD 2023.10.22 (git)

Parameters
- $fa: 4
- $fs: 0.25

Export
- Base filename: export
- generatedxf: ✓
- generategcode: ✓
- generatesvg: ✓

Feeds and Speeds
- feed: 400
- large V ratio: 0.875
- plunge: 100
- small V ratio: 0.75
- speed: 16000
- square ratio: 1

Joinery
- Glue Gap: 0.05
- Joinery Gap: 0.1

CAM
- large ball tool no: 0
- large square tool no: 0
- large V tool no: 301
- small ball tool no: 0
- small square tool no: 102
- small V tool no: 390
- toolradius: 1.5875

Stock
- retractheight: 9
- stocklength: 219
- stockorigin: Lower-Left
- stockthickness: 7.9
- stockwidth: 150
- zeroheight: Top

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 (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
- pygcodepreview.scad (pyscad) --- the Python functions wrapped in OpenSCAD
- gcodepreview.scad (gcpscad) --- OpenSCAD modules and variables
- gcodepreview_template.scad (gcptmpl) --- example file

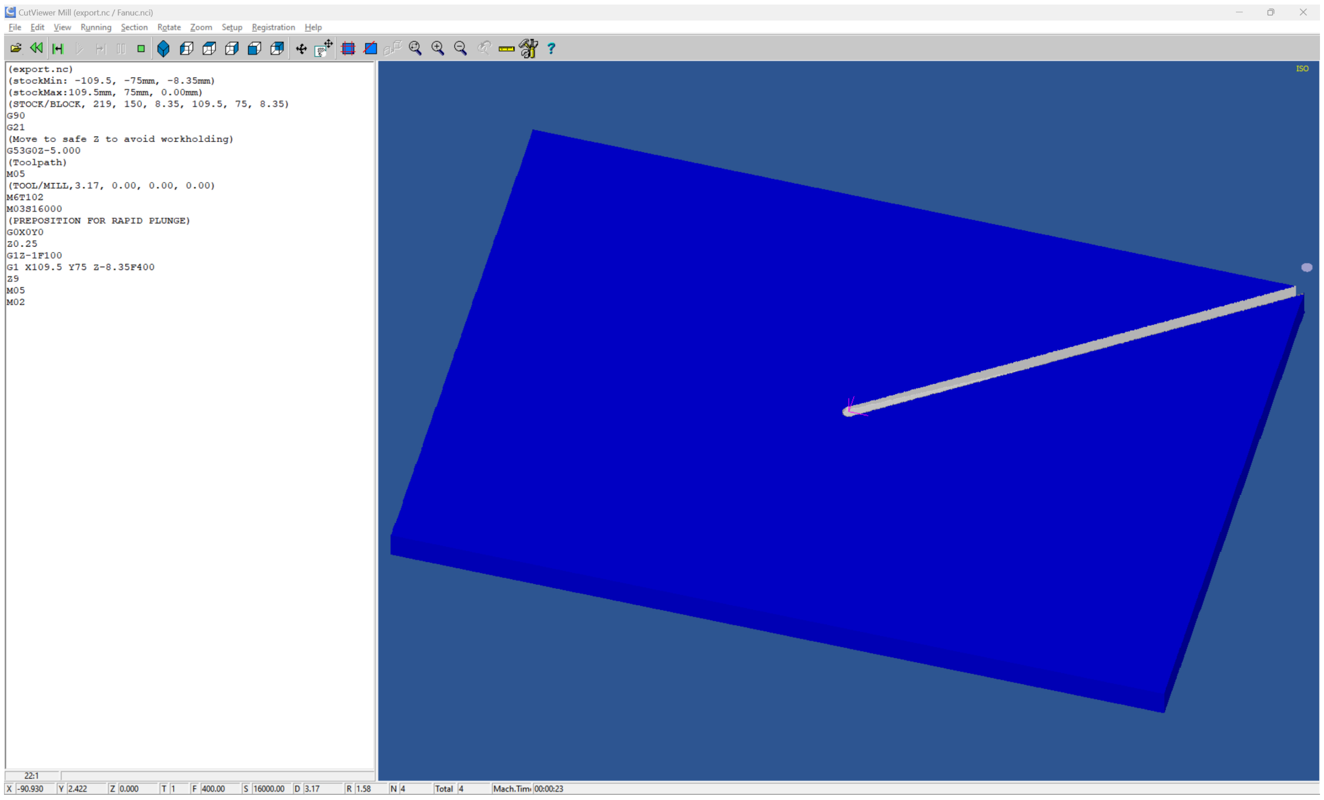
- cut2Dshapes.scad (cut2D) --- code for cutting 2D shapes

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:


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

```
(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
```

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).

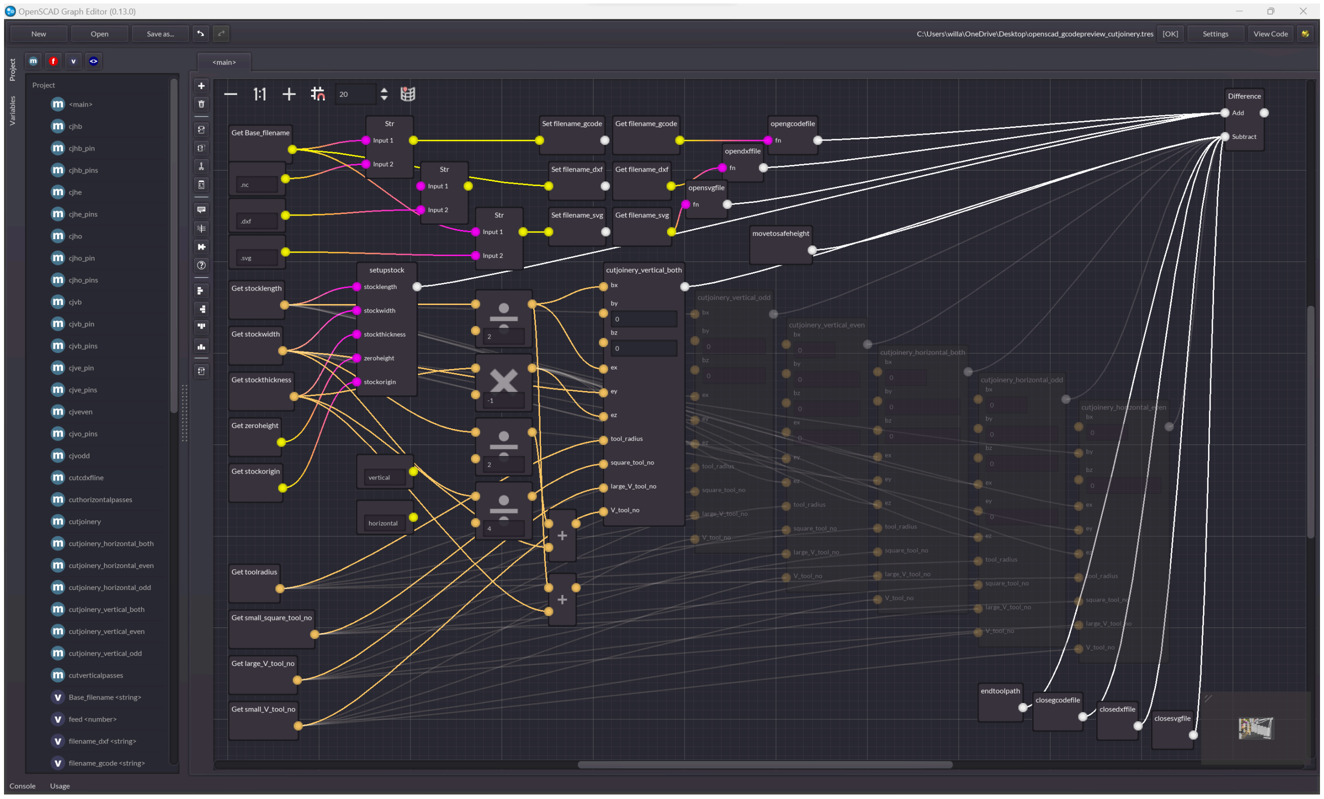
Importing this DXF and actually cutting it is discussed at:

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

Version 0.3

- Support for curves along the 3rd dimension
- support for roundover tooling

Version 0.4

- Rewrite using literati documentclass, suppression of SVG code
- dxfrectangle (without G-code support)

Version 0.5

- more shapes
- consolidate rectangles, arcs, and circles in gcodepreview.scad

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
- additional generalized modules for cutting out various shapes/geometries

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 would be written out upside down due to coordinate system differences between OpenSCAD/DXFs and SVGs requiring managing the inversion of the coordinate system (it is possible that METAPOST, which shares the same orientation and which can write out SVGs 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. ↩

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>; use <pygcodepreview.scad>; include <gcodepreview.scad>; ↩