Contents

1 Blender4CNC

1.1 Blender4CNC - A Blender Addon

1.1.1 Introduction

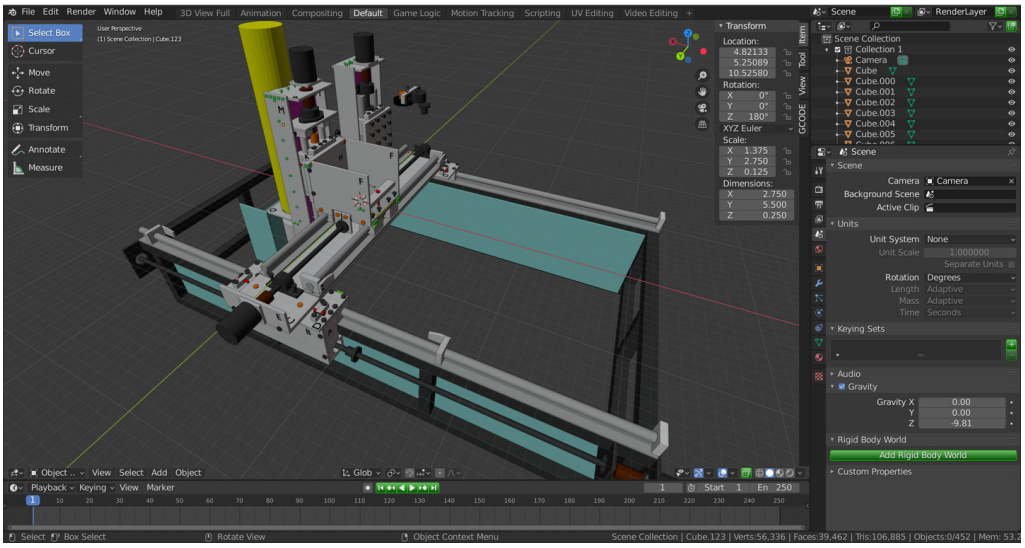

Figure 1: Blender is a very powerful open-source 3D modelling software application. It is free for download and runs under Linux, Mac OS and Windows. It is well suited to visualizing all things 3D.

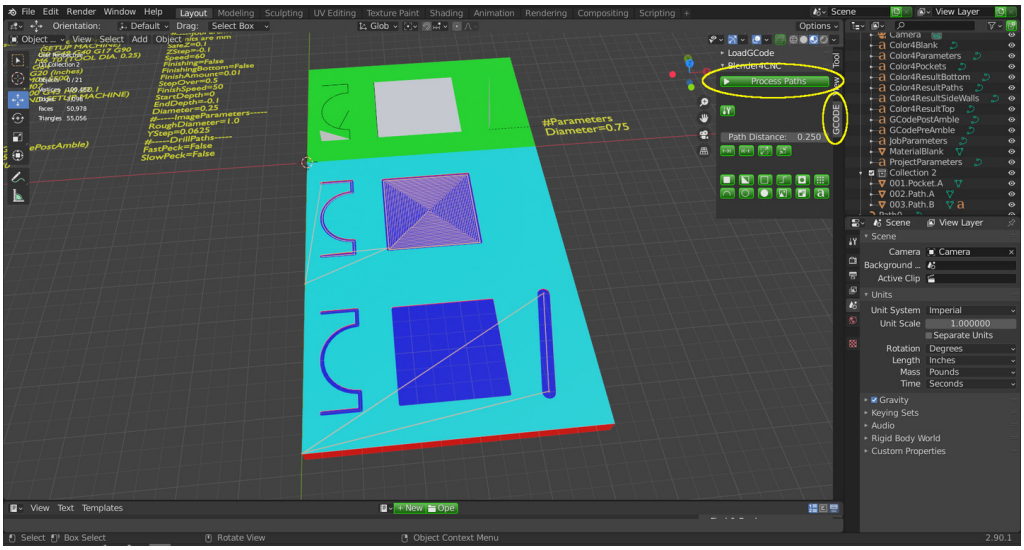

Figure 2: Blender4CNC is a Blender Addon/Plugin to produce GCode for routing a design on a 3-axis CNC machine. It can be downloaded at `https://github.com/ddommett/Blender4CNC`. It allows a user to draw tool paths as simple objects. These paths can be modelled and subtracted from a block of material to visualize the end result while producing GCode to send to the CNC machine. Skip to the section "Installing Blender" if you're ready to get started.

1.1.2 Why develop Blender4CNC?

I was constantly looking for open source software for CNC maching. A search on the internet will find numerous exemplary applications (ranging from free to expensive) and I think that is fantastic. However, I wanted software that would be open source and free forever; that would run on any operating system; that would manage curves exactly; and that would be simple to install and use. Every software package required trade-offs. Besides, I wanted the answer to be Blender!

Traditionally, CNC maching exists in the world of CAD/CAM. CAD (computer aided design) is a software tool used to create digital 2D and 3D drawings to design a variety of items. Computer models are created and defined by geometrical patterns. CAM (computer aided manufacturing) uses geometrical design data to control automated machinery. These systems often use knowledge of CNC machines and tools to convert a 3D design into GCode. A CNC machine can "run" the GCode to produce the designed part. For decades, the industry has strived to achieve a work flow whereby a part is designed and visualized (without any concerns of HOW it is to be manufactured) and then at the press of a "magic" button it can be manufactured. (CAM software is quite complex and requires information about one or more CNC machines in order to perform this duty.) It was never my intention to try and compete with this level of complexity.

I was a CNC hobbyist. I only had one machine and I just wanted to see that machine move. I wanted to see it just do something! Anything really, just simple stuff. I didn't have an automatic tool changer or even many cutters from which to choose. And it was just a straight forward 3-axis CNC machine which meant that most of what it did was - move "up" in Z, move over to an X,Y start location, move down in Z, cut a path or pocket in 2-dimensions (X,Y) and then raise up again. I know the machine I am going to use for EVERYTHING I design and I know which bit I will be using. I don't need complicated software, I mainly need software that can say - "Move over here and cut a shape."

Therefore, the Blender4CNC plugin requires a user to specify and draw paths for their machine to follow. Blender4CNC enables some very complicated and accurate shapes to be produced with (hopefully) minimal effort for the user.

I tried to make sure the plugin was simple to use and easy to install on multiple operating systems.

But it is not easy to achieve these goals perfectly. Afterall, Blender is proudly a "mesh" oriented 3D tool aimed at being used by professionals to make movies and video games. It knows it's users and wants to be the best at what it does for those users. It is unashamedly NOT a CAD application. A CAD application will handle mathematical calculations with great accuracy and precision. Blender is not inaccurate but, for example, when a user is generating a 3D model of a character they usually don't care if the tip of the foot is **exactly** at 12.73952". If the tip lands at 12.739" (or even 12.74" rounded up) it is probably going to look just fine. Also, CAD applications usually allow a user to generate objects "parametrically" and to allow for numerous mathematical shapes (like exact curves for example). Blender does not define exact curves but yet CNC machines and GCode **do** use exact curves. While Blender does have such things as bezier curves and NURBS curves, neither of these are intended for defining exact curves. Most of the 3D modelling capabilities of Blender revolve around using a "mesh" (see figure 3). A mesh is a set of points in space (sometimes millions of points) representing any complex shape like a human body for example. The points are connected with edges, usually to generate many triangles (or quads) which are called faces. Any curved surface is approximated by a set of linear segments and this looks great when you use sufficient points/edges/faces and is ideal for use in video games and movies. So how do you represent maching operations and paths with meshes in Blender?

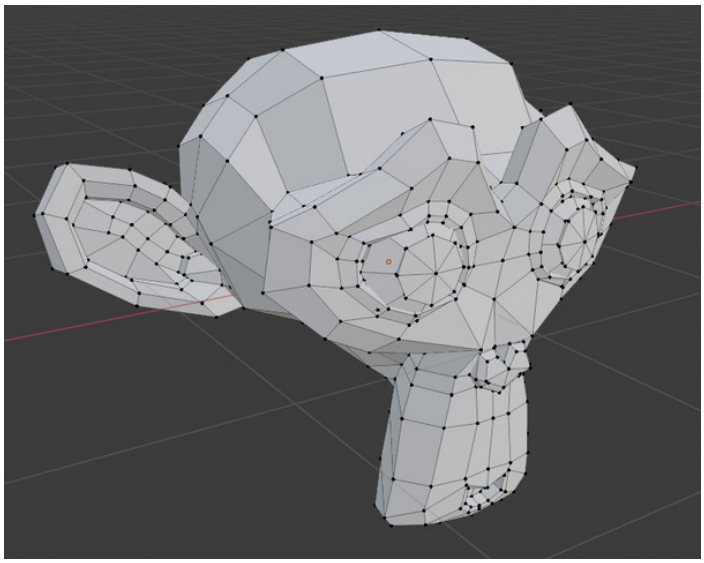

Figure 3: A monkey face designed in Blender as a mesh. Most of the faces in this example are "quads" not triangles.

The best I could come up with (eventually) was to use certain conventions and two dimensional meshes. I gave specific names to machining operations like "Pocket" or "Path". I gave specific "attributes" on segments in a mesh to indicate the origin and direction of a maching operation. How do you represent an "exact" curve (like a G2 or G3 G-Code)? While Blender does have "curve" objects, it has no concept of mixing straight line segments with curved segments in a mesh. (And things like a spline curve are a real pain to use to represent circular arcs.) I felt it was important to be able to represent **exact** arcs amongst straight line segments when defining a geometrical shape. This is so that the G-Code produced can be as mathematically exact as possible to the resolution of the machine hardware. (Of course, nothing stops a person from approximating a curve with a series of small, straight line segments if they choose.)

While developing Blender4CNC I have had to wrestle with the accuracy and precision of floating-point computations. This is a common issue that can affect all kinds of software that requires accuracy and precision. In a later section I talk a little about this issue and the trade-offs. I think I have largely succeeded in dealing with most possibilities but I cannot guarantee that there are not bugs of this kind waiting to be found.

Blender Addons are programmed in the Python language and this affects performance. While complicated operations can produce GCode quite quickly, the code to visualize the operations in 3D takes significantly longer to run. I have tried to profile the code and eliminate bottlenecks and improve algorithms. The nature of python is such that I cannot easily run multiple threads to make use of multiple cores or even use multi-processing to spawn multiple processes as Python is embedded within Blender. I am trying to avoid writing high-performance code in compiled C because that would bring with it a plethora of issues for building from source code on different architectures and operating systems and this would complicate user installation. Maybe in the future I can readdress the issues of performance. Maybe WSL on Windows 10 would allow a build to be simply limited to Linux but used on multiple OSes. Maybe the addon could shell out to Python outside of Blender and spawn multiple processes or a "server" to perform calculations. For now, the performance "is what it is".

One final comment: Blender is VERY powerful and packed with many features I have never even used. It should be possible for a user of Blender4CNC to customize their CNC projects by using what Blender already includes. It is possible to display their material with a texture (maybe a wood grain like pine or oak or walnut etc. or aluminum, black steel etc.) A user can render their output visualization to an image. They might want to generate an animation from it. They may want to add their own annotations to the screen. I feel that Blender provides a framework for a CNC addon to continue to grow in ways that I cannot imagine right now.

1.1.3 Installing Blender

Blender4CNC **requires** Blender version 3.4.1 or later. The website for Blender is www.blender.org. There are numerous different Linux distributions which have slightly different install tools. Some will have a fairly recent version of Blender in their repositories and it can be installed easily via a "Software

Manager" application. Some will recommend using "Snap" to install Blender. You can also download Blender from the main website or even download the source code and build it yourself. Go ahead and install Blender for your operating system.

1. Installing Blender on LinuxCNC (and possibly other specialized Linux distributions)

   These instructions are valid as of February 10, 2023 for LinuxCNC 2.8.0 and Blender version 3.4.1.

   If you are using a very specialized distribution of Linux (such as LinuxCNC) then the following may be helpful for installing Blender. Download the latest version of Blender from www.blender.org. (At the time of writing this was blender-3.4.1-linux64.tar.xz.) Open a terminal in the location where the file was saved and unpack the file with a command like:

   ```
   tar -xvf blender-3.4.1-linux64.tar.xz
   ```

   You can then run Blender by changing into the new folder and executing it:

   ```
   cd blender-3.4.1-linux64
   ./blender
   ```

1.1.4 Installing Dependencies

Blender4CNC **requires** libraries to be available to Blender. Blender makes use of the "Python" language and environment and we must make sure that this Blender-Python environment can access the libraries: Numpy, Scipy and Scikit-image. This step is a little different for each operating system. Firstly, we need to know where the Blender-Python environment is installed.

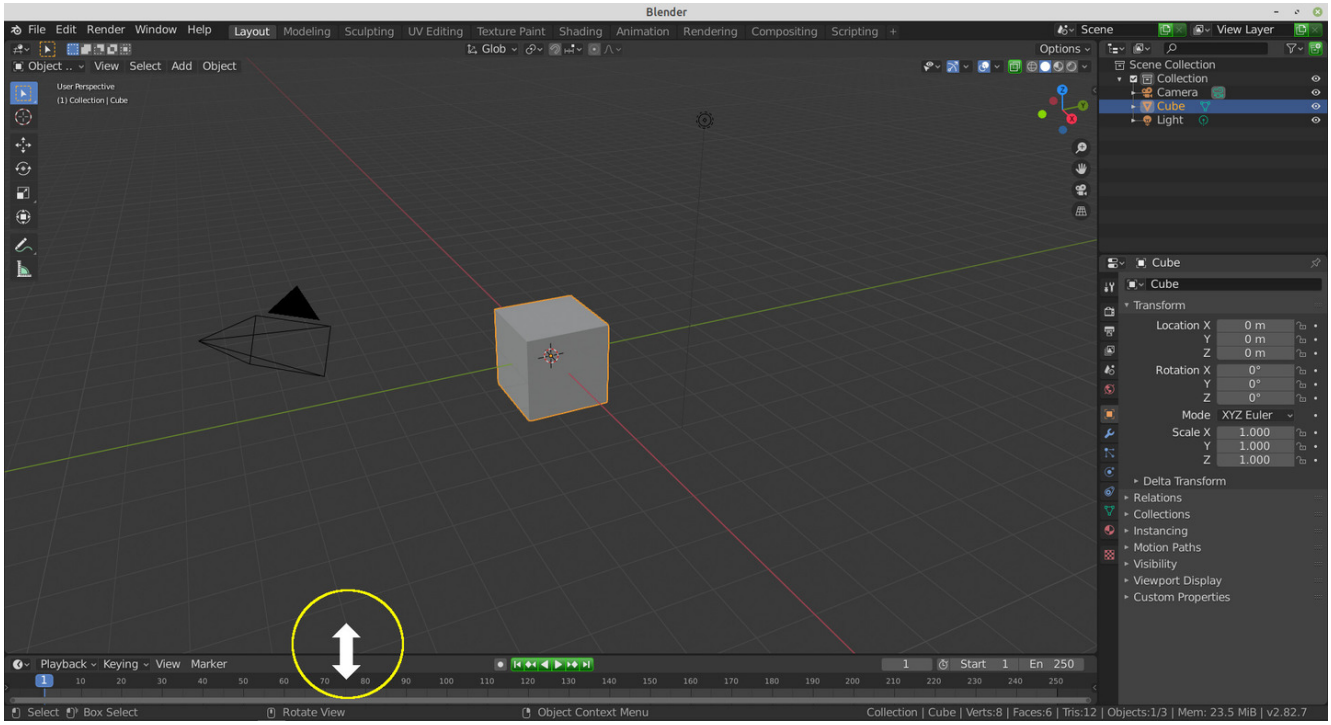

Figure 4: Open a "Python Console" in Blender. If Blender has just been installed, it likely looks like this figure and you can just enlarge the lower window by moving the mouse between the two sub-windows until it changes to a double-ended arrow and dragging the edge of the window up. If there is just one large window (like the 3D view window) then create a 2nd window by moving the mouse to the lower edge of the main window until you see the double arrow; right-click the mouse and select "Horizontal Split". Then drag the lower window up to make it larger.

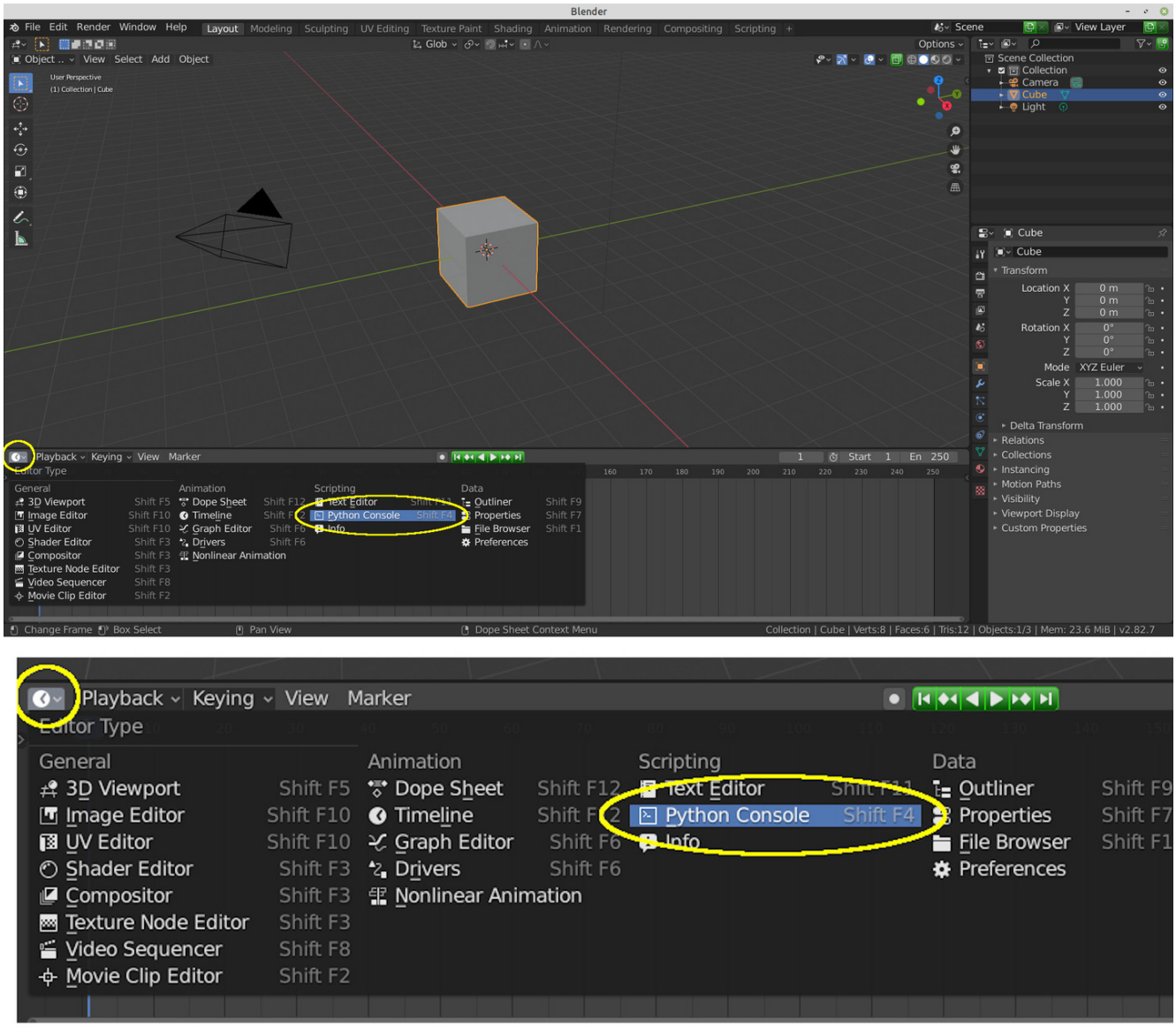

Figure 5: Click on the "Editor Type" menu button in the upper left corner of the lower window and select "Python Console".

Then type the following commands at the python console prompt (the triple ">" prompt) (figure 6):

```
import sys
sys.exec_prefix
```

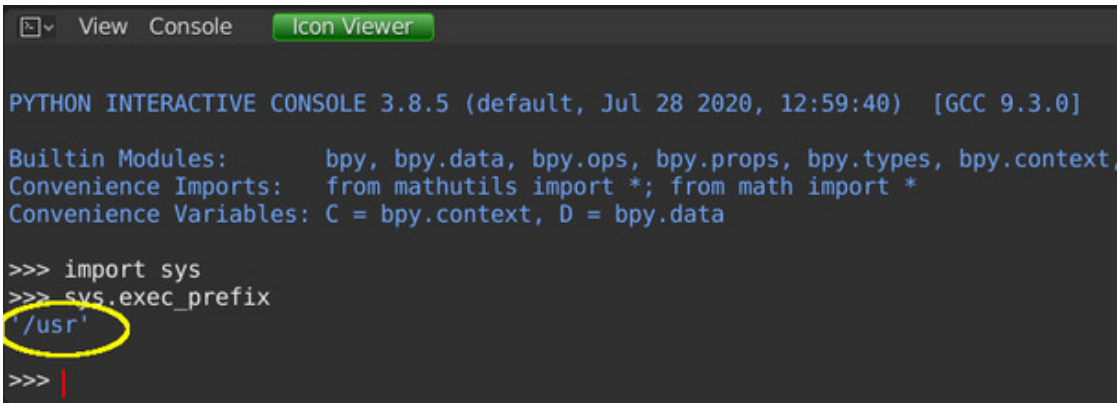

Figure 6: The Blender-Python location

If you have specially installed Blender 3.4.1 into a special folder in Linux you may see something like:

```
'{PATH TO INSTALL}/blender-3.4.1-linux-x64/3.4/python'
```

If you are fortunate to use a Linux distribution that contains Blender 3.4.1 or later in it's repositories, you may see:

```
'/usr'
```

If using Windows, it is likely that you will see something like:

```
C:\\Program Files\\Blender Foundation\\Blender 3.4\\3.4\\python
```

Remember this folder name where Blender Python is installed. Jump to the following section which matches the operating system you are using.

1. Installing Dependencies in Windows

   These instructions are valid as of February, 2023 for Windows 10 and Blender version 3.4.1.

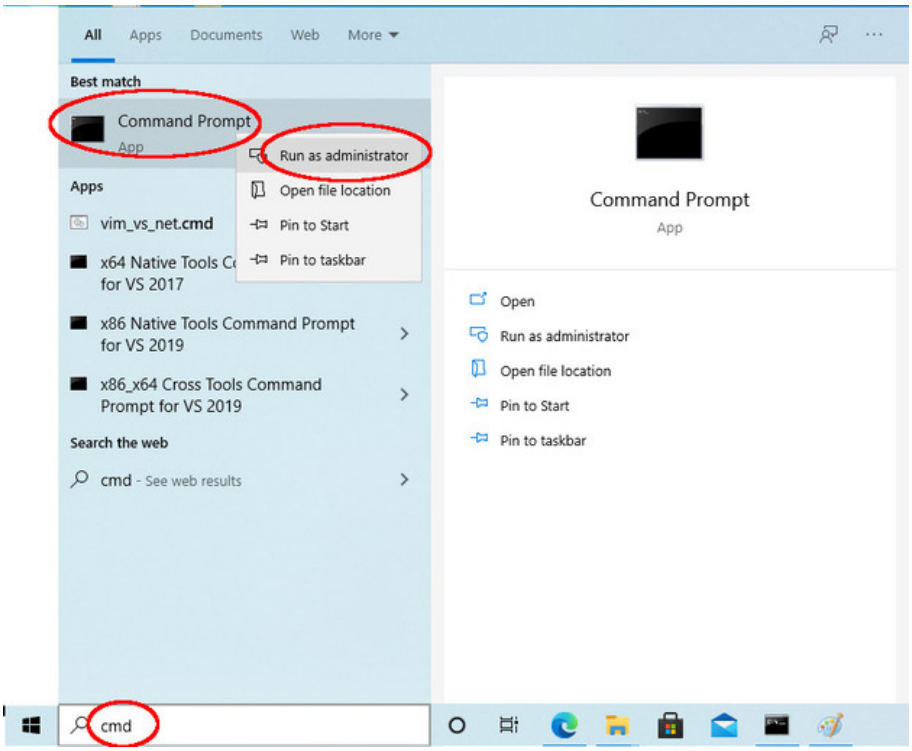

Figure 7: Open a Console "Command" window **in Administrator Mode**. In the search field, type "cmd", then right-click on the "Command Prompt" application and select "Run in Administrator Mode". A terminal prompt will open.

You will have to change the directory (a "cd" command) to where Blender-Python is installed. For example (only need single slashes):

```
cd "C:\Program Files\Blender Foundation\Blender 3.4\3.4\python"
```

Then type the following command to install the libraries:

```
.\bin\python.exe -m pip install scipy scikit-image
```

This might take a few minutes as you will see the process download files (figure 8) then the libraries will be installed.

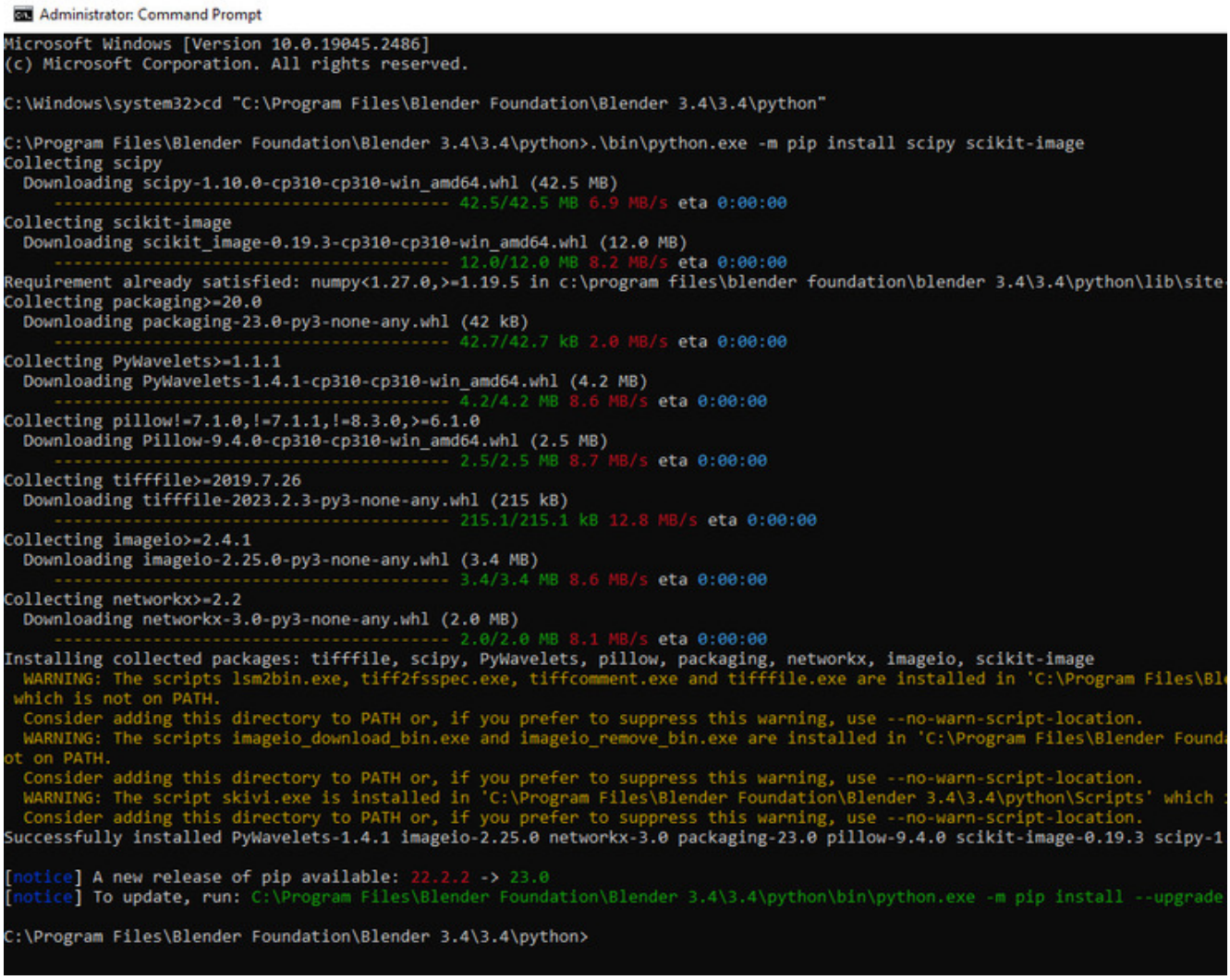

Figure 8: Installing Scipy and Scikit-image

2. Installing Dependencies in Linux (Ubuntu-based e.g. Linux Mint)

   These instructions are valid as of February, 10 2023 for Linux Mint 20.1 and Blender version 3.4.1.

   Open a terminal and type (pip may already be installed but this will make sure, you may be prompted to enter an administrator password):

   ```
   sudo apt-get install python3-pip
   ```

   If your Blender-Python location that you noted previously is something general like "/usr" then Blender is using the version of python that came with Linux and you can just type the following command:

   ```
   pip3 install scipy scikit-image
   ```

   Otherwise, if you are working with a special version of Blender that has been installed in a non-standard location, then you must change to the correct folder and type the following commands (remember to substitute the real folder name you noted previously in place of {folder} and don't include the curly brackets e.g. cd /Downloads/blender-3.4.1-linux-x64/3.4/python):

   ```
   cd {folder}
   ./bin/python3.10 -m ensurepip
   ./bin/pip3 install scipy scikit-image
   ```

   (Different versions of Blender have used different names for the python executable in the "bin" folder. You may have to check the exact name. I have seen things like: python3, python3.7m, python3.10 etc.)

3. Installing Dependencies in OpenSUSE Linux

   If your Blender-Python location is something general like "/usr" then Blender is using the version of python that came with Linux and you can just type the following command:

   ```
   sudo pip3 install scipy scikit-image
   ```

   Otherwise, if you are working with a special version of Blender that has been installed in a non-standard location, then you may have to change to a different folder and type slightly different commands (see the section under installing in Ubuntu based Linux systems for suggestions).

4. Installing Dependencies in Fedora Linux

   If your Blender-Python location is something general like "/usr" then Blender is using the version of python that came with Linux and you can just type the following command:

   ```
   sudo dnf install scipy scikit-image
   ```

   Otherwise, if you are working with a special version of Blender that has been installed in a non-standard location, then you may have to change to a different folder and type slightly different commands (see the section under installing in Ubuntu based Linux systems for suggestions).

5. Installing Dependencies in LinuxCNC

   Open a terminal and type (pip may already be installed but this will make sure, you may be prompted to enter an administrator password):

   ```
   sudo apt-get install python3-pip
   ```

   You must change to the correct folder and type the following commands (remember to substitute the real folder name in place of {folder}):

   ```
   cd {folder}
   ./bin/python3.7m -m ensurepip
   ./bin/pip3 install scipy scikit-image
   ```

1.1.5 Installing Addon

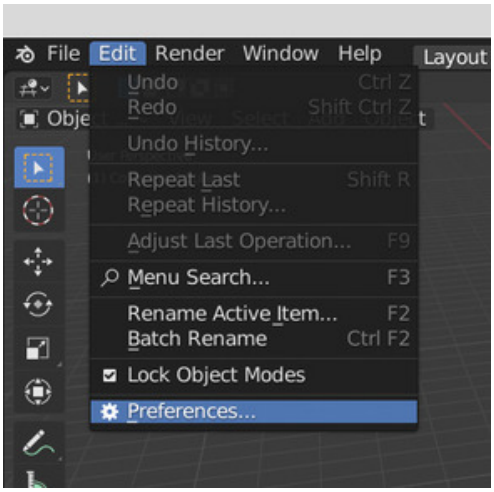

Figure 9: Download the zip file containing Blender4CNC: Go to `https://github.com/ddommett/Blender4CNC` and under the "Code" pulldown, select the option to download a zip file. This zip file contains the addon, templates and example projects and documentaion as pdf files. Unzip this file somewhere on your computer. To install the addons in Blender, go to the main Blender window, "Edit" -> "Preferences".

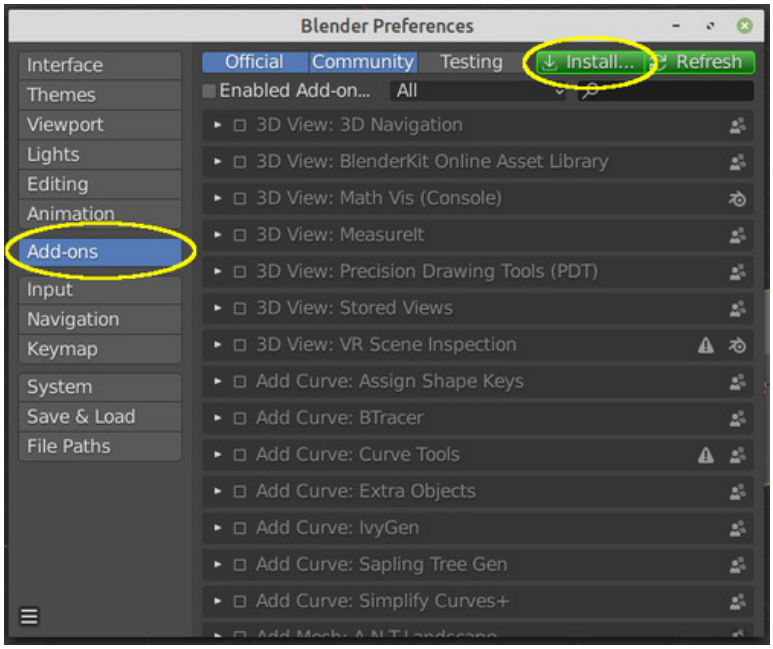

Figure 10: Click "Add-ons" and then click "Install" and browse to where you have the "Blender4CNC.py" file.

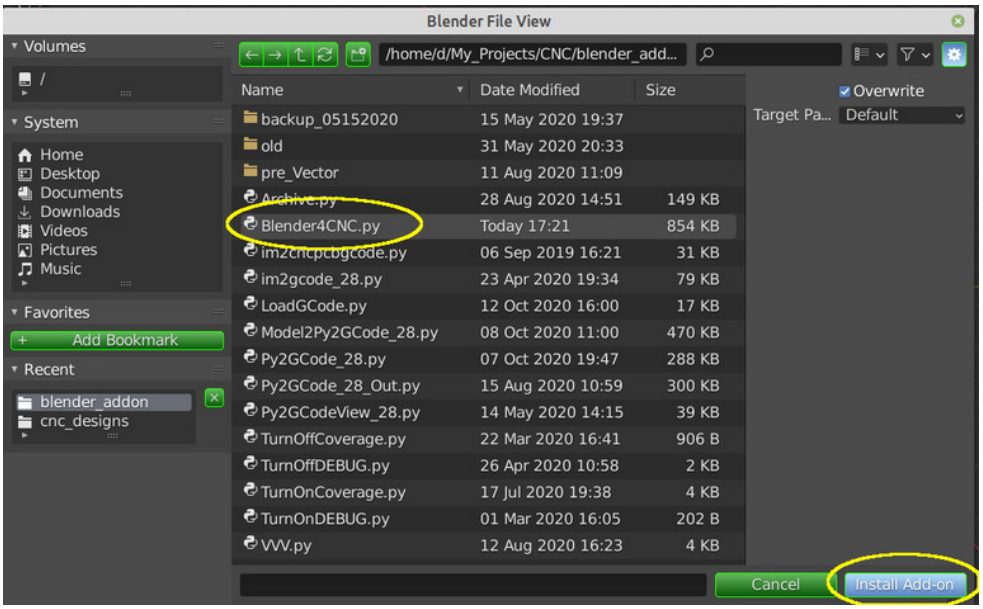

Figure 11: Select the "Blender4CNC.py" file then click "Install Add-on".

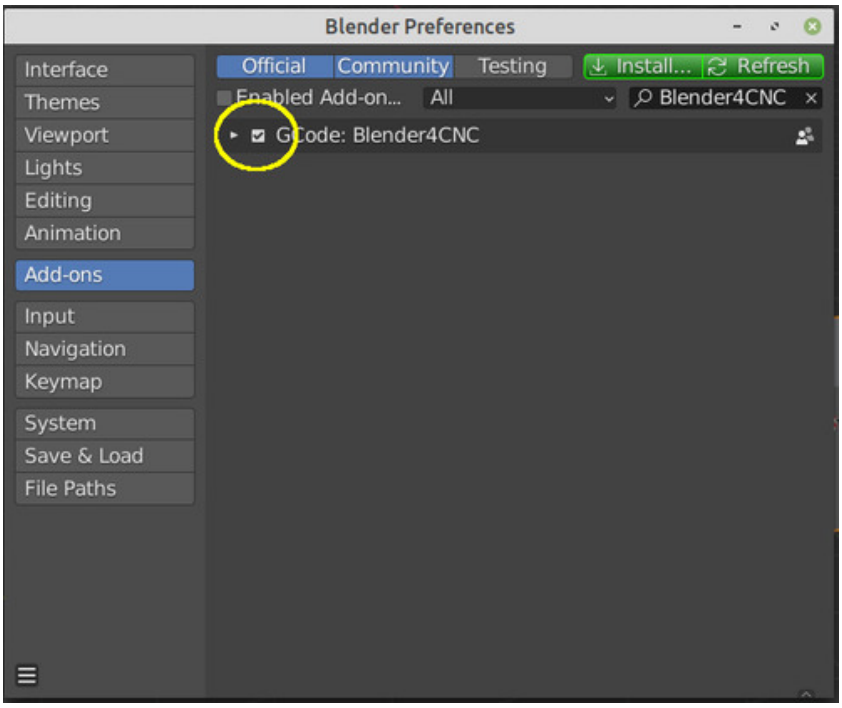

Figure 12: Enable the Addon by checkmarking the box. You may need to scroll down the list of many addons to see the Blender4CNC addon you have just installed. (You can also trim the list by clicking the little arrow in the filter box that says "All" and selecting GCode.)

Close the Preferences window.

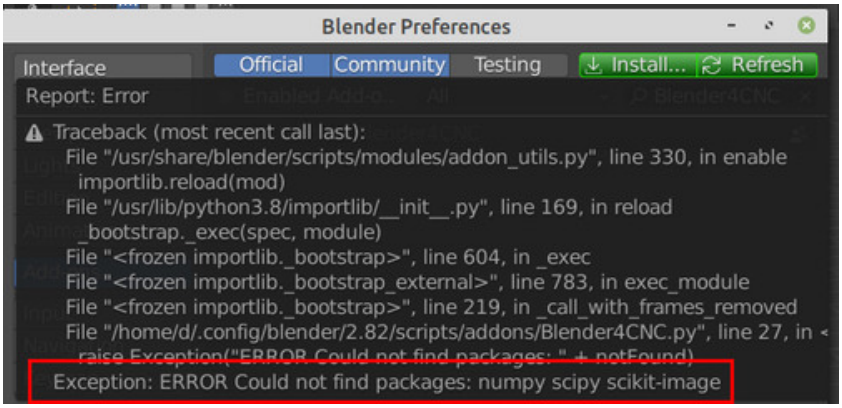

Figure 13: An error occurs if the packages - Numpy, Scipy, and Scikit-image - are not installed in the version of Python that Blender uses. You must install the dependencies (see previous section).

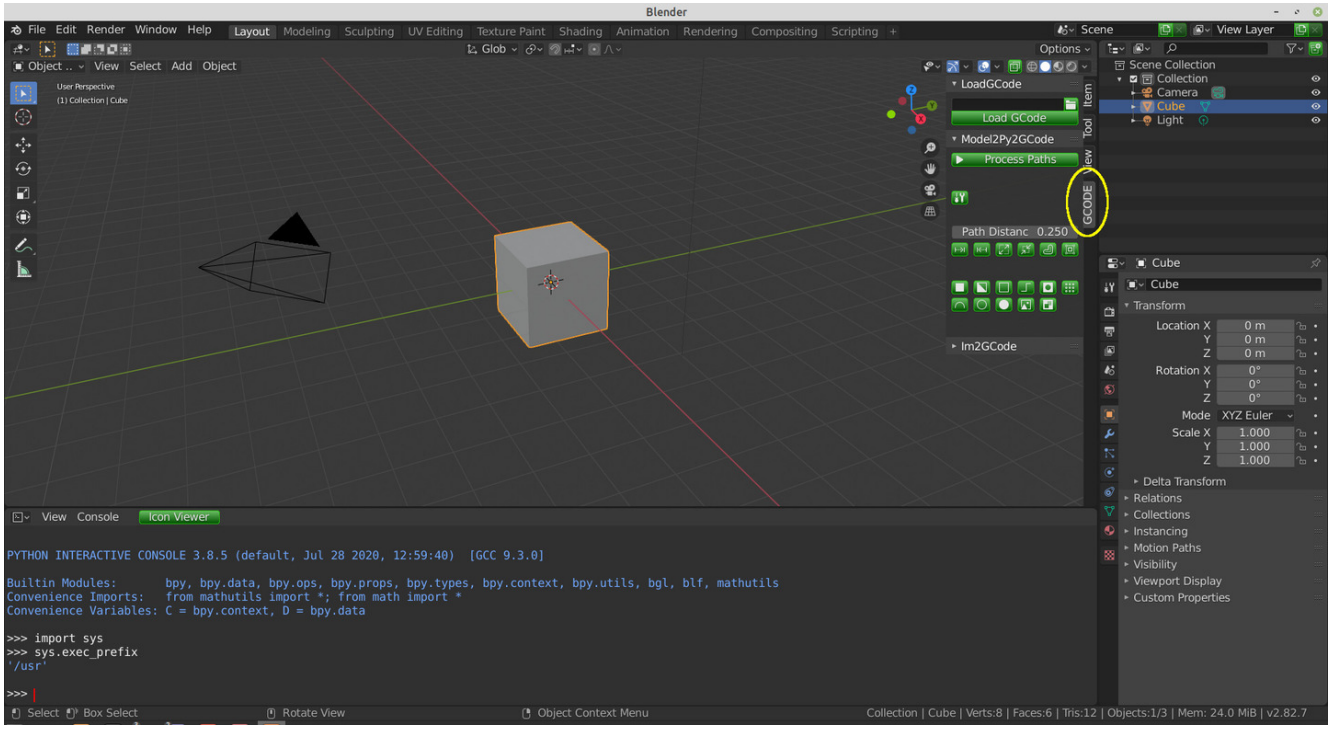

Figure 14: If all is successful you will see a new GCode tab on the right edge of the viewport. **If you don't see the GCode tab (or any tabs) it might be that the panel is not visible - click the tiny little arrow at the top right of the viewport shown in figure 15).**

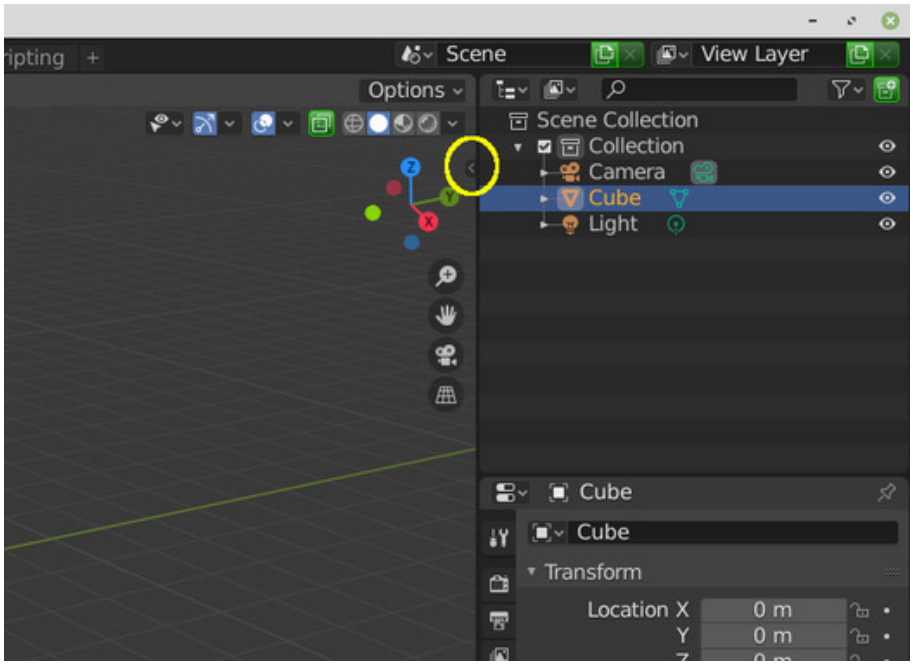

Figure 15: Click tiny arrow to make panel visible.

1. Other useful Add-ons (optional)

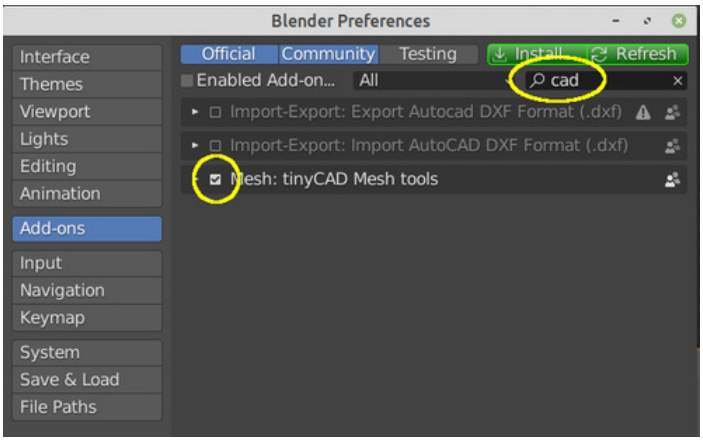

Figure 16: Another useful addon is "tinyCAD". It is not necessary but I find it useful on occasion. Prior to Blender version 2.80, it was necessary to download tinyCAD separately but now it seems to be packaged with Blender. This means that you can simply enable it if you choose. Pull up the "Edit" -> "Preferences" window and enter "cad" into the search field, then checkmark the box to enable Mesh:tinyCAD Mech tools. Then close the Preferences window.

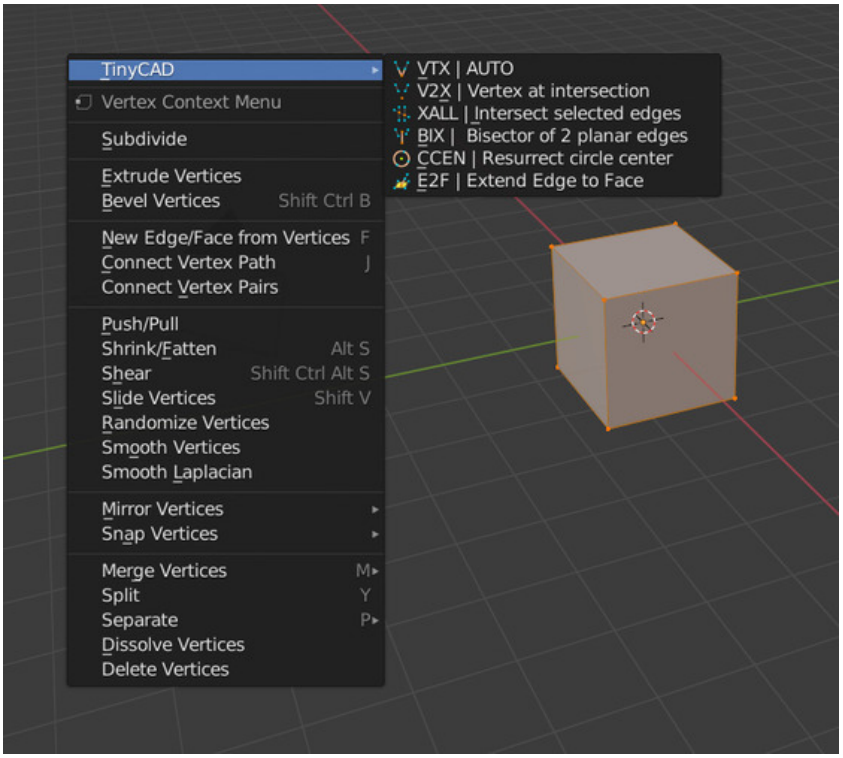

Figure 17: This shows an example of the tinyCAD menu. It provides a few useful functions for creating intersections. It appears when in Edit Mode and you right-click in the viewport. Don't worry if you are not familiar with Edit Mode - that will be demonstrated in other sections.

1.1.6 Configuring Blender (Optional)

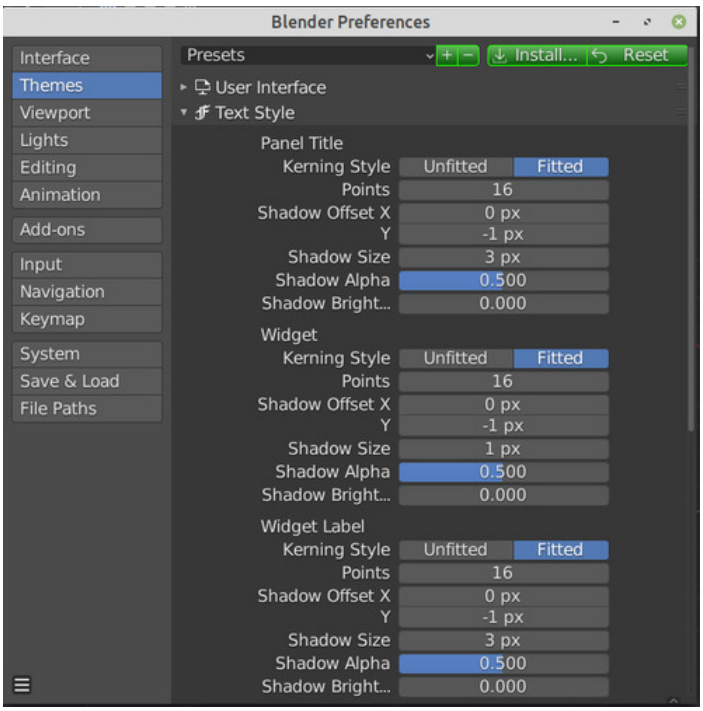

Figure 18: I like to configure the UI of Blender slightly. I set the Text to size 16.

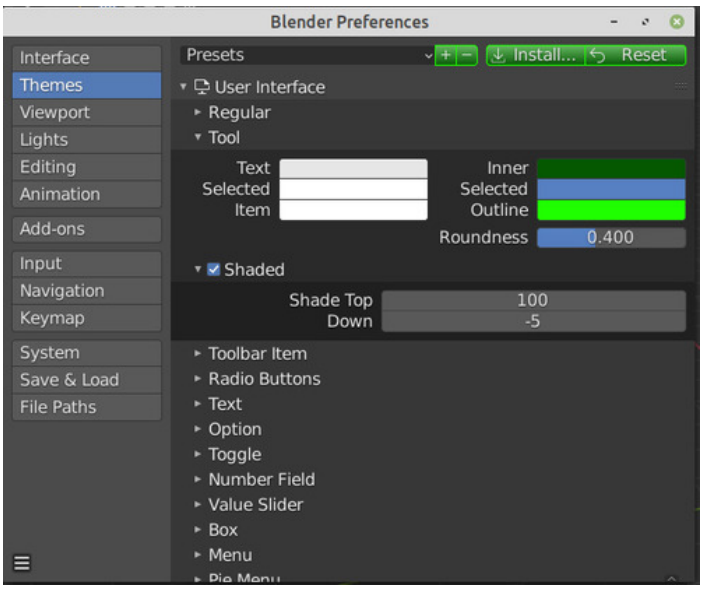

Figure 19: I also like to make buttons bright green as this is easier for me to see amongst the many fields and functions on a very busy Blender screen. On a side note, humans see green better and with less eye strain than any other color and this is why early monochrome monitors were green and why night vision goggles also originated with green phosphor for display. (The "inner" is set to RGB values of 0.125, 0.5 and 0. The "outer" is set to RGB values of 0.125, 0.9 and 0.)

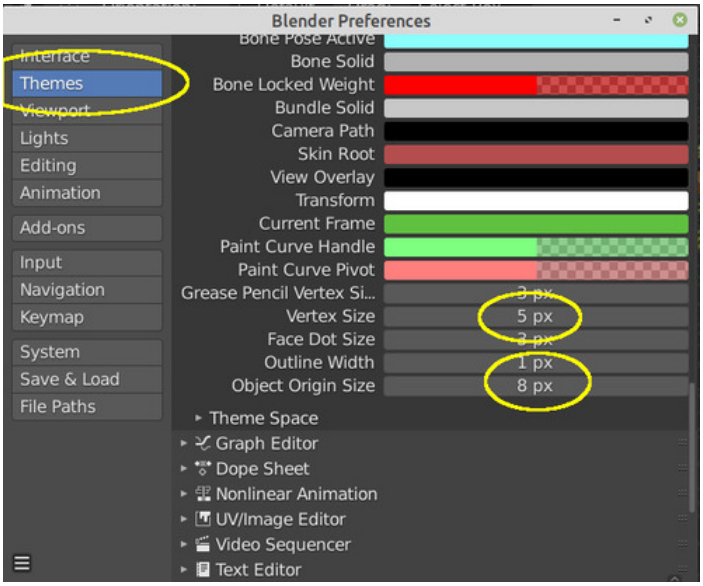

Figure 20: I also like to make the vertices a little larger sometimes (This is in Themes->3D Viewport.).

1.2 Getting Started - OverView

This section is a brief overview of the Blender4CNC addon. It shows a basic CNC project. Later sections will explain projects and features in more detail. Every CNC project needs some basic things - default parameters, starting and ending GCode snippets (which are specific for each CNC machine setup) and routing operations to produce the design.

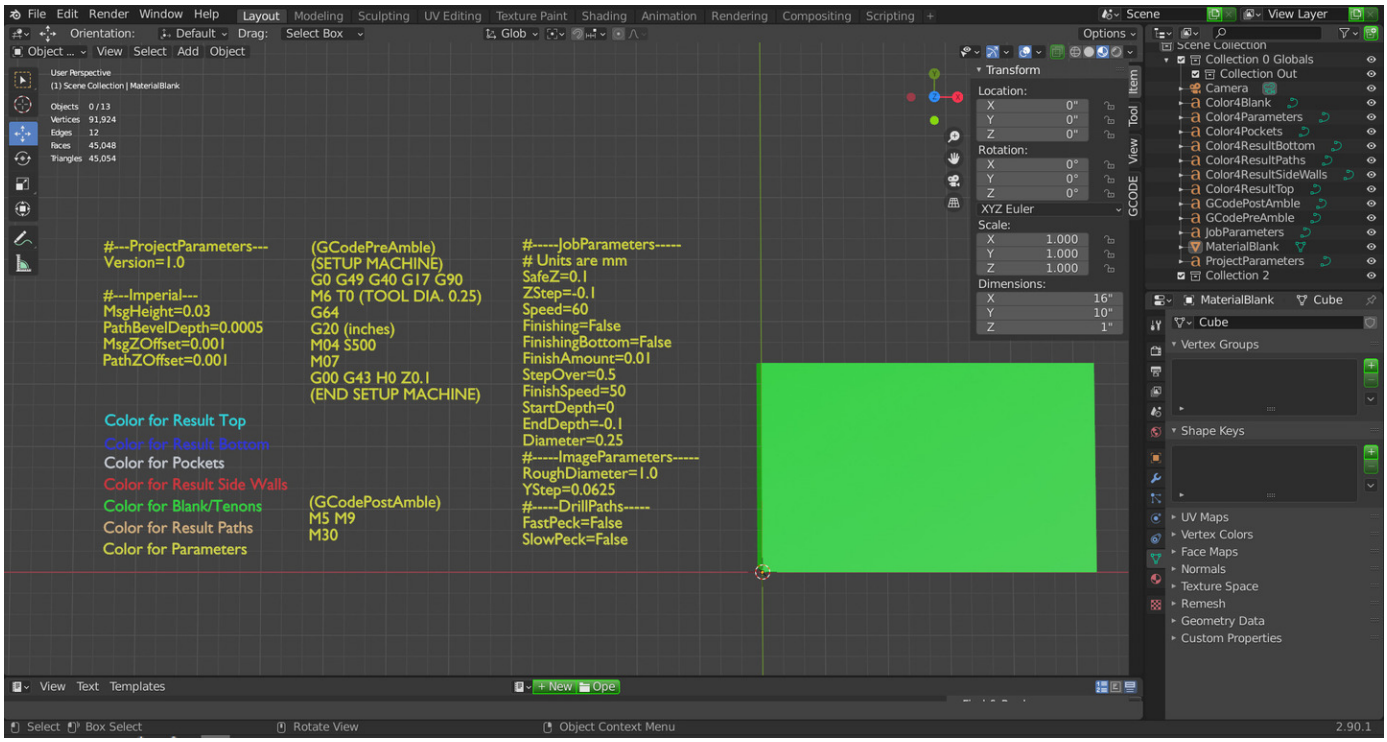

Figure 21: Material "Blank" (a block of raw material).

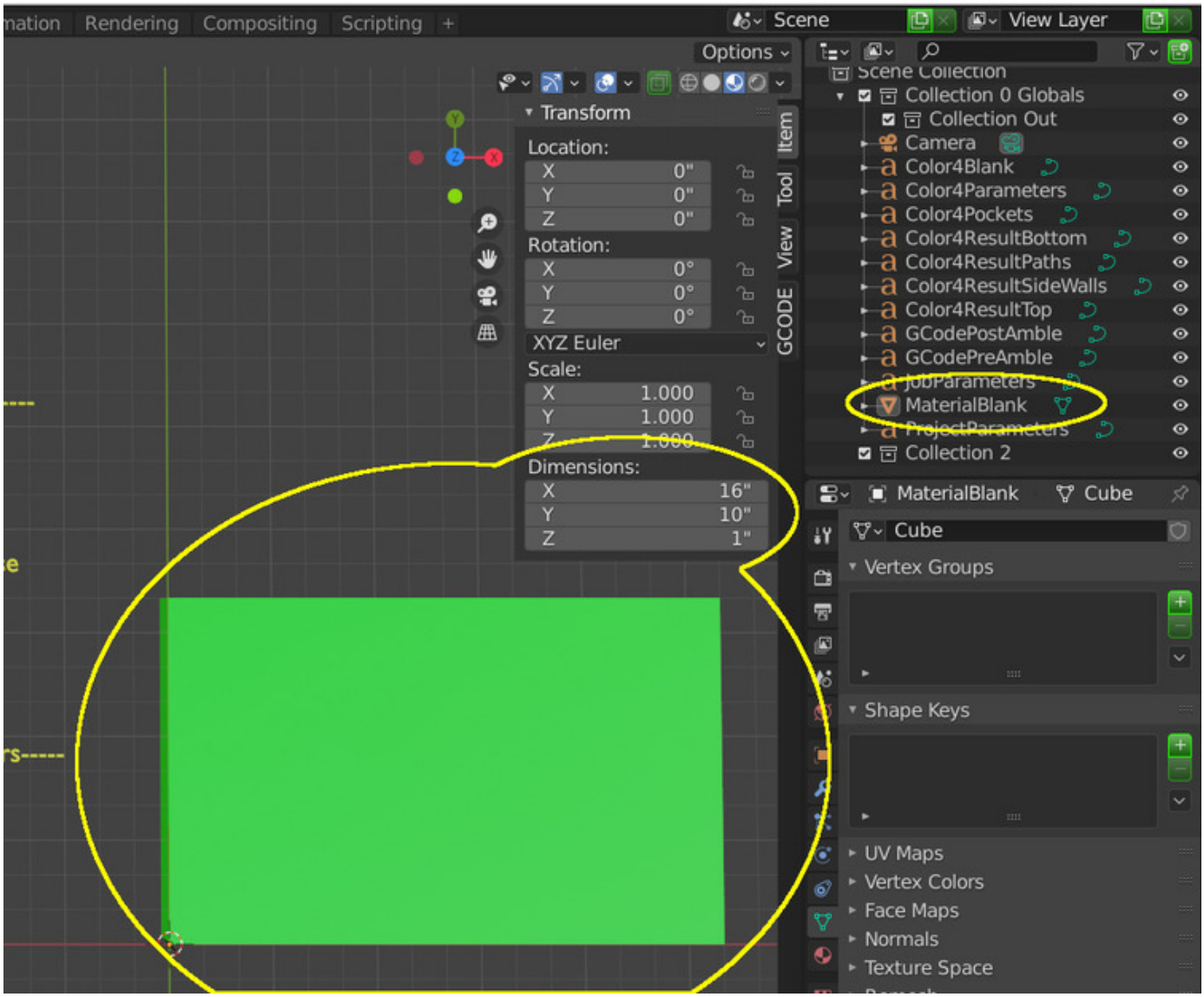

Figure 22: A CNC project needs a "blank". This is our block of material into which we will carve. It **must** be called "MaterialBlank". The size has been set to 10"x6" and it is 1" in height in this example.

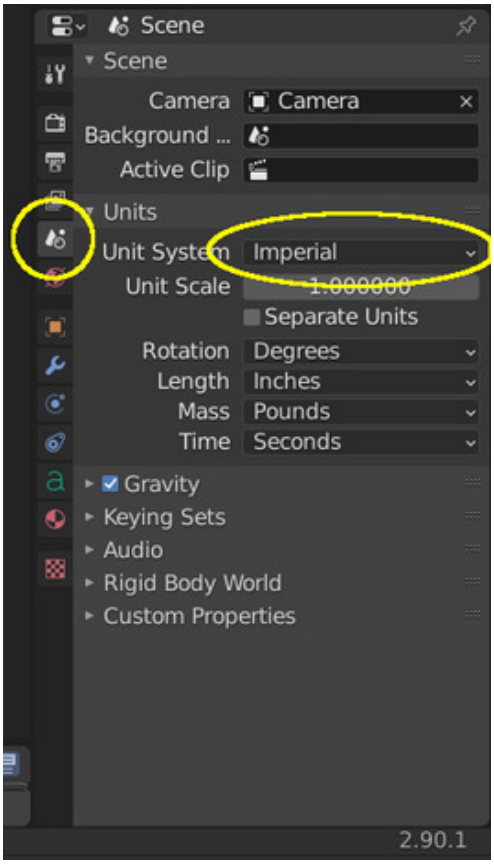

Figure 23: In Blender, in the lower right corner, units can be set in the "Scene Properties". Blender4CNC **requires** it be set to "Imperial" or "Metric".

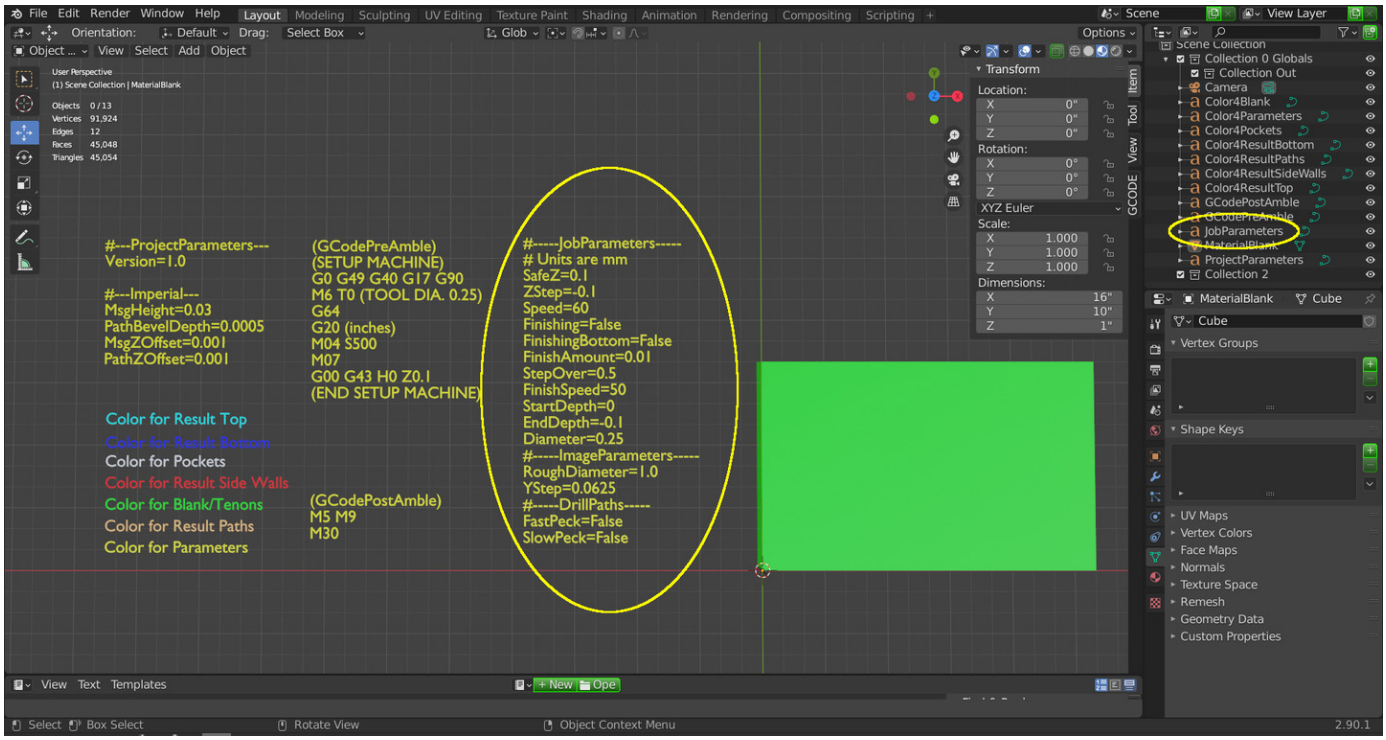

Figure 24: Default job parameters are defined in a text object which **must** be named "JobParameters". Any line of text that starts with a "#" is a comment (I like to put the name of the text object in a comment at the top so I can see it easily). This defines the default parameters for the job but any CNC operation can override these parameters as necessary.

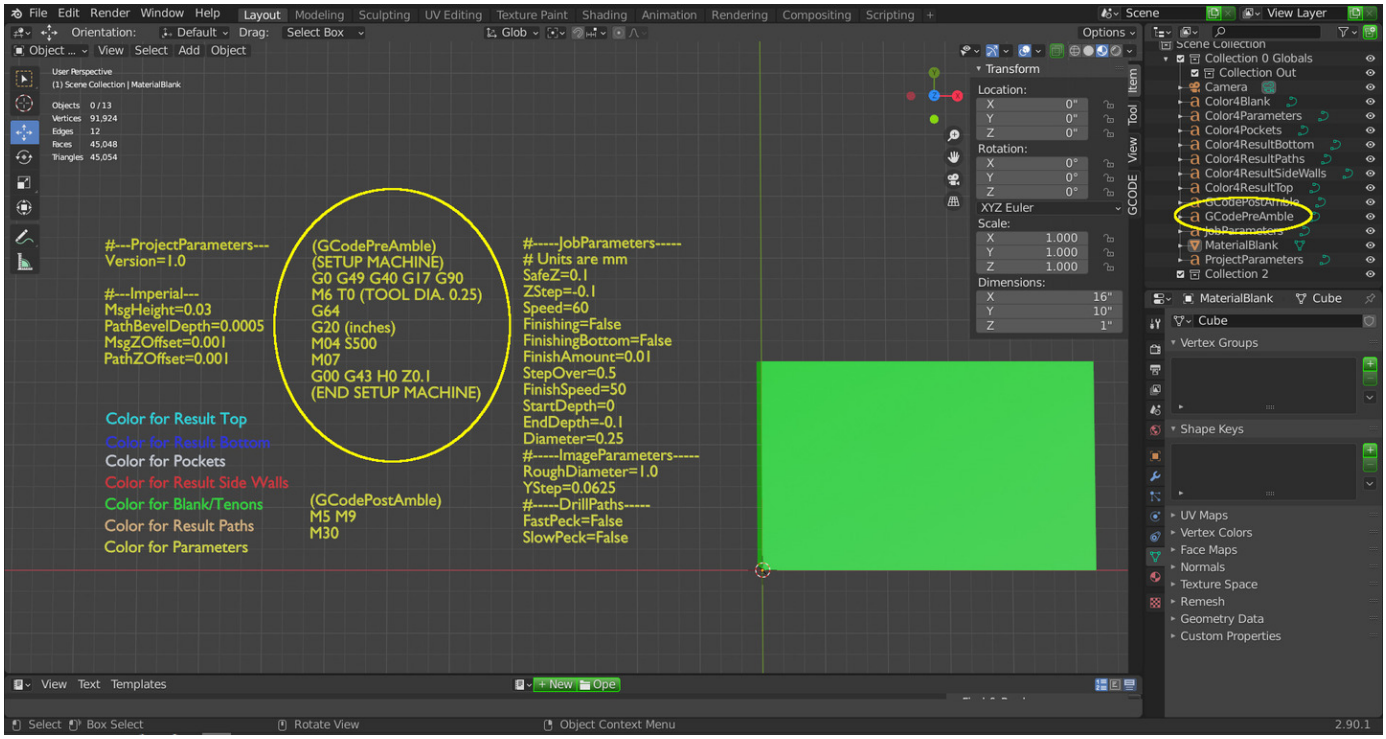

Figure 25: The GCode PreAmble is a text object which **must** be called "GCodePreAmble". It is a simple block of GCode text that will be inserted at the beginning of any GCode produced from this project. It is specific to the CNC machine being used and usually sets up the machine to the default modes used most often. On the first non-comment line in this example, G49 ensures any tool offsets are cancelled; G40 cancels any compensation; G17 selects the X-Y plane; and G90 specifies absolute positioning. These commands and the subsequent lines cancel any remembered settings from previous jobs and setup the machine for the current job.

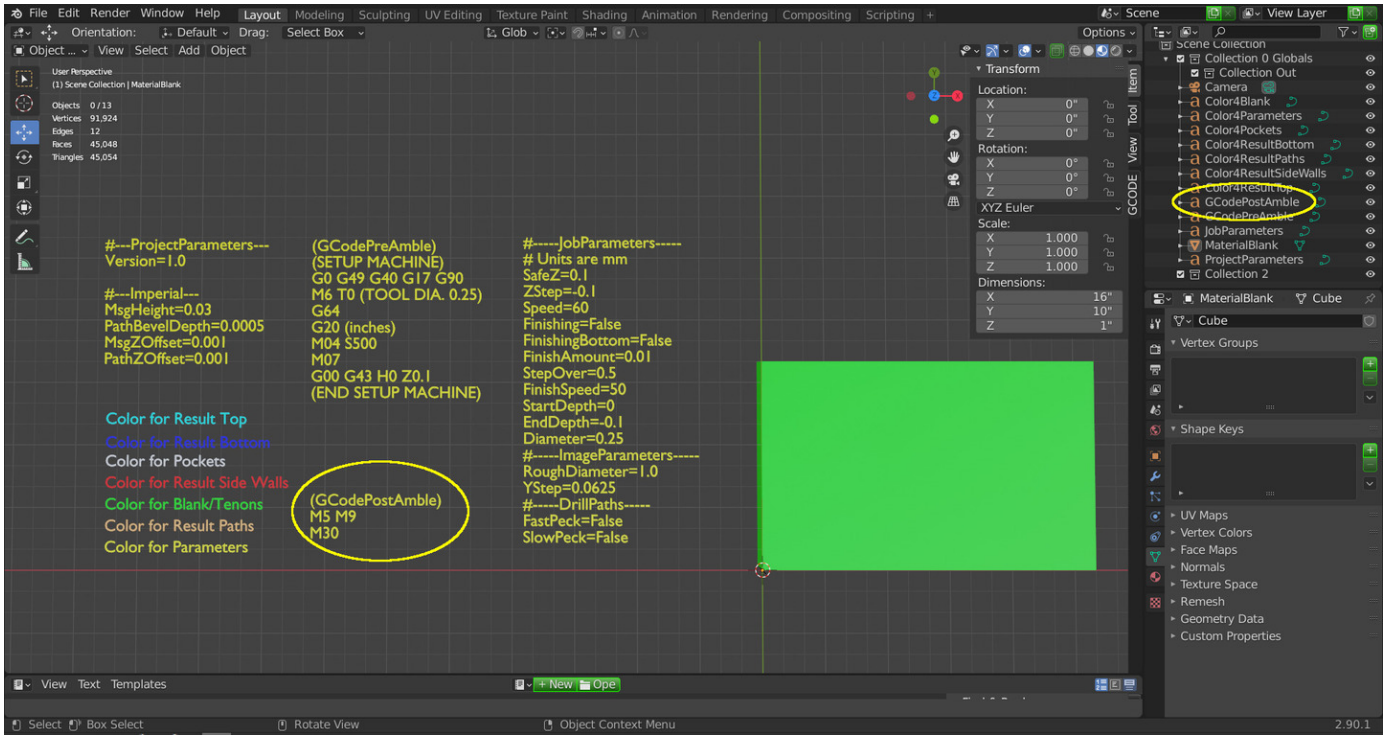

Figure 26: The GCode PostAmble is a text object which **must** be called the "GCodePostAmble". It is a simple block of GCode text that will be inserted at the end of any GCode produced. It is specific to the CNC machine being used and usually just safely ends the program. In this example, M5 will stop the spindle; M9 turns off coolant; and M30 ends the program. (In my machine, the spindle isn't controllable from GCode so the M5 does nothing but it doesn't hurt either. I don't hane coolant either but I use the "coolant" controls to enable power relays for dust collection so M9 will turn the dust collector off at the end.)

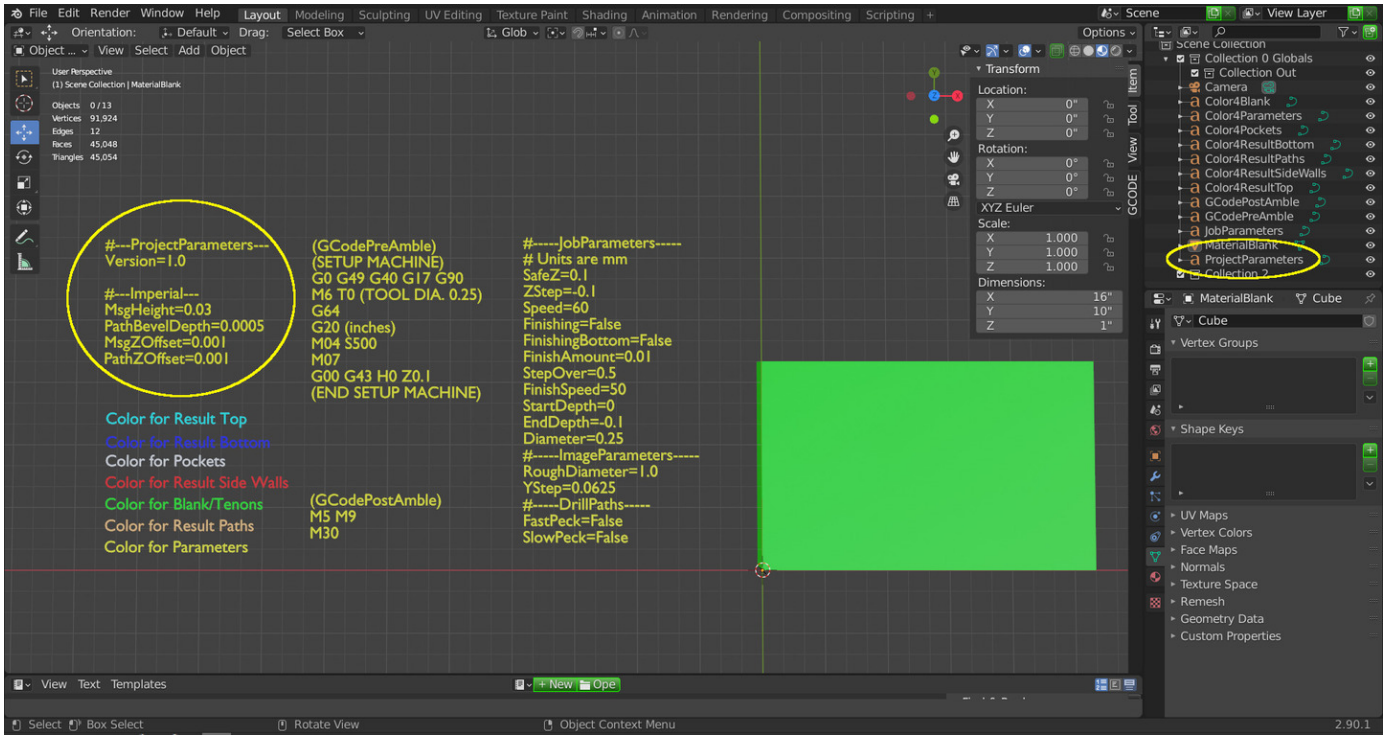

Figure 27: The Project Parameters is a text object which **must** be called "ProjectParameters". It sets some parameters that are used project-wide (particularly for dealing with the difference between Imperial and Metric units).

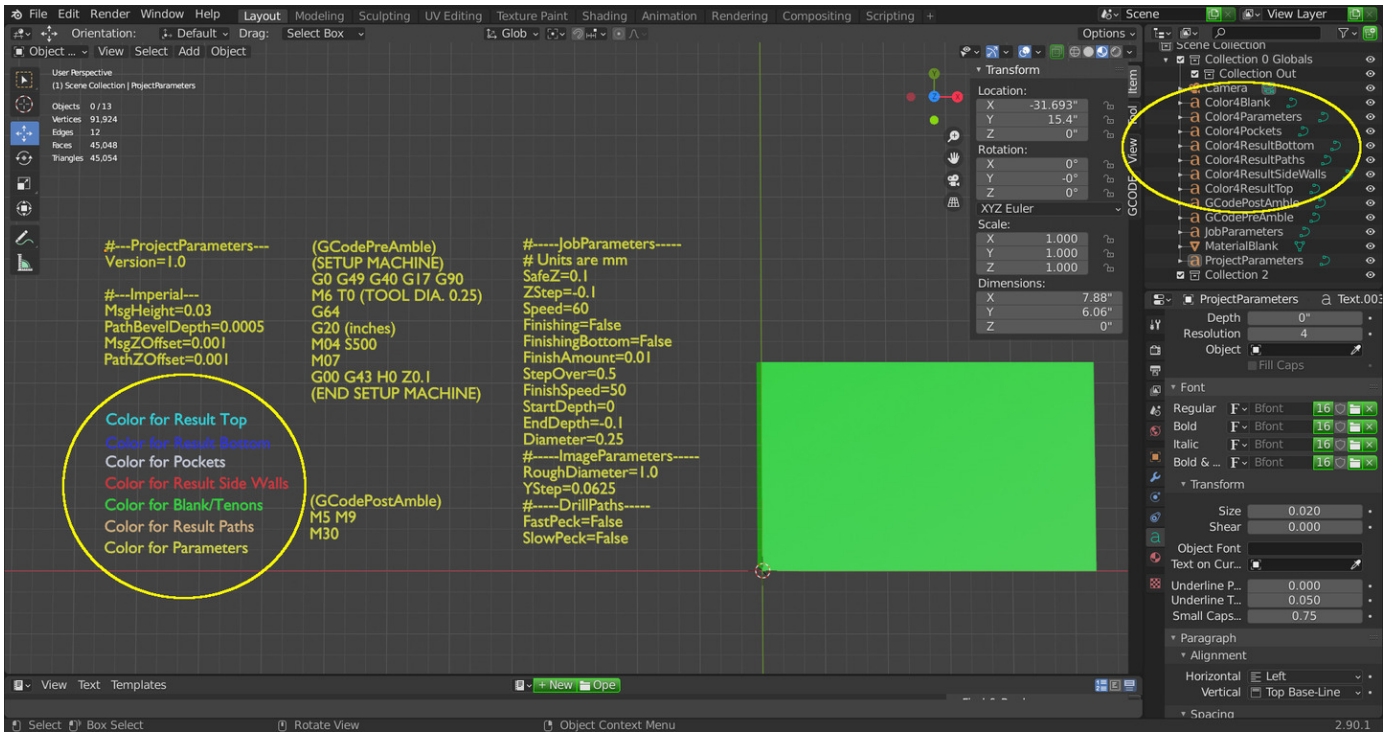

Figure 28: Optionally, a project contains a set of objects to allow a user to configure preferred colors by selecting a text object and changing the color of the material.

The project consists of a series of operations. All operation names **must** follow a naming convention: {3-digit Number}.{Type}.{Comment}. An example name might be: "010.Pocket.SquareHole". The 3-digit number specifies in what order the operations occur (i.e. 001 will be cut before 002 etc.). The "Type" specifies the type of operation which is a pocket in this case. The comment can be any text you wish. Table 1 outlines the different types of operations with examples of the naming convention. As long as the job parameters are specified, you just draw the operation and Blender4CNC will calculate the GCode.

Table 1: Table of Operations

Type	Example Name	What it Does
Path	010.Path.Comment	Cuts a path (like a line)
Pocket	020.Pocket.Comment	Cuts an area (like a square hole)
DrillPath	030.DrillPath.Comment	Drills many holes
Hole	040.Hole.Comment	Drills a single hole
DepthImage	050.DepthImage.Comment	Engraves a 3D "image" from a grayscale image
Tenon	060.Tenon.Comment	Leaves an "island" of material inside a pocket

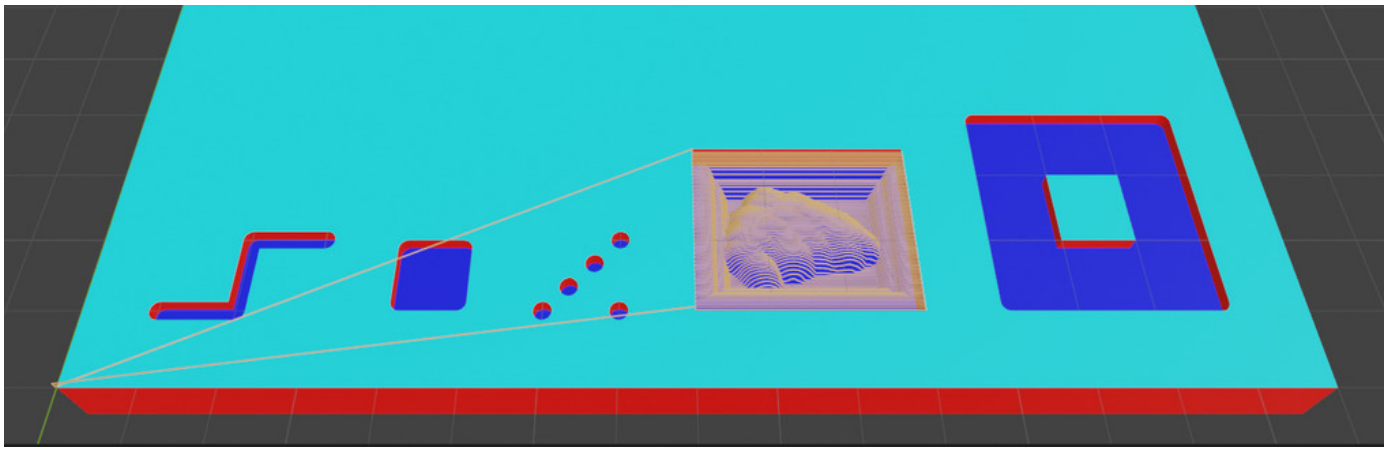

Figure 29: Examples of Types (left to right: Path, Pocket, Drill Path, Hole, Depth Image, Tenon).

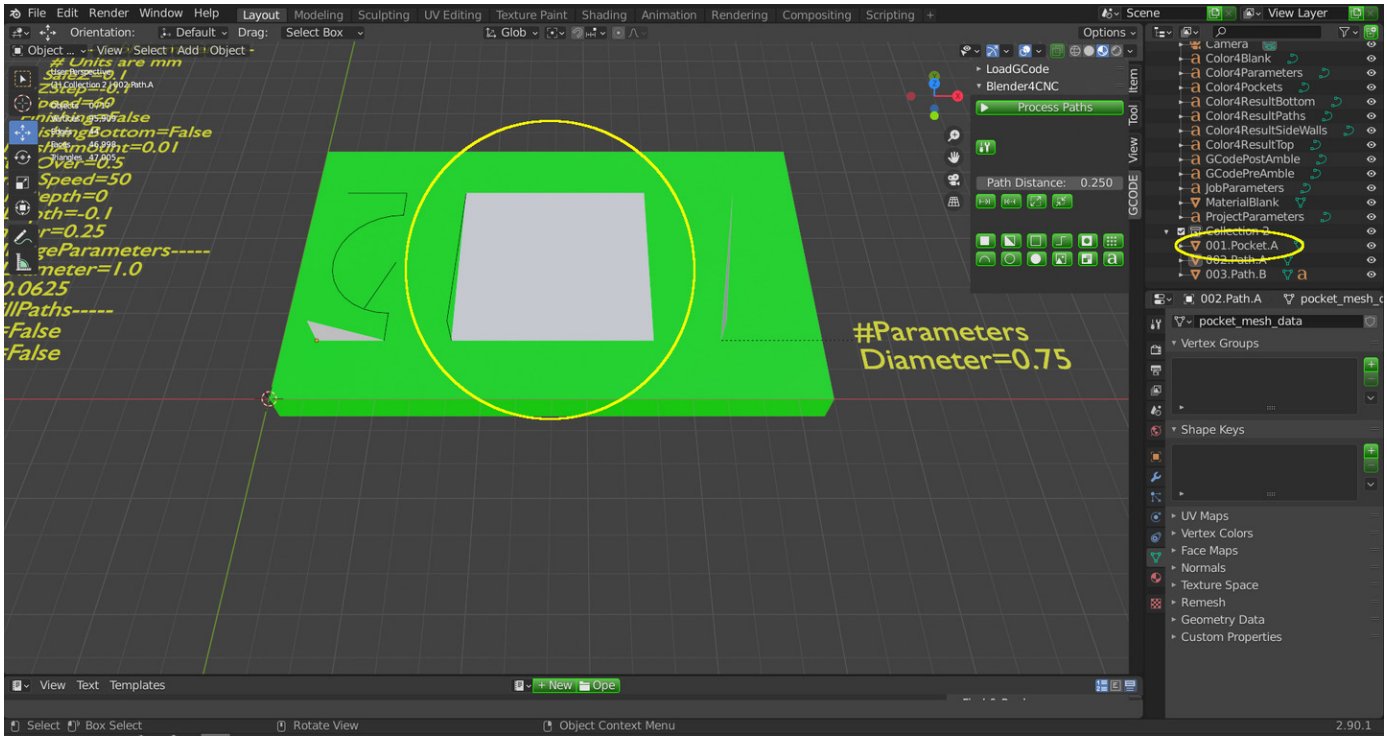

Figure 30: Here is a project with three operations. The first operation is a pocketing operation called "001.Pocket.A". A pocketing operation tells the CNC machine to remove all the material inside the boundaries of the pocket shape.

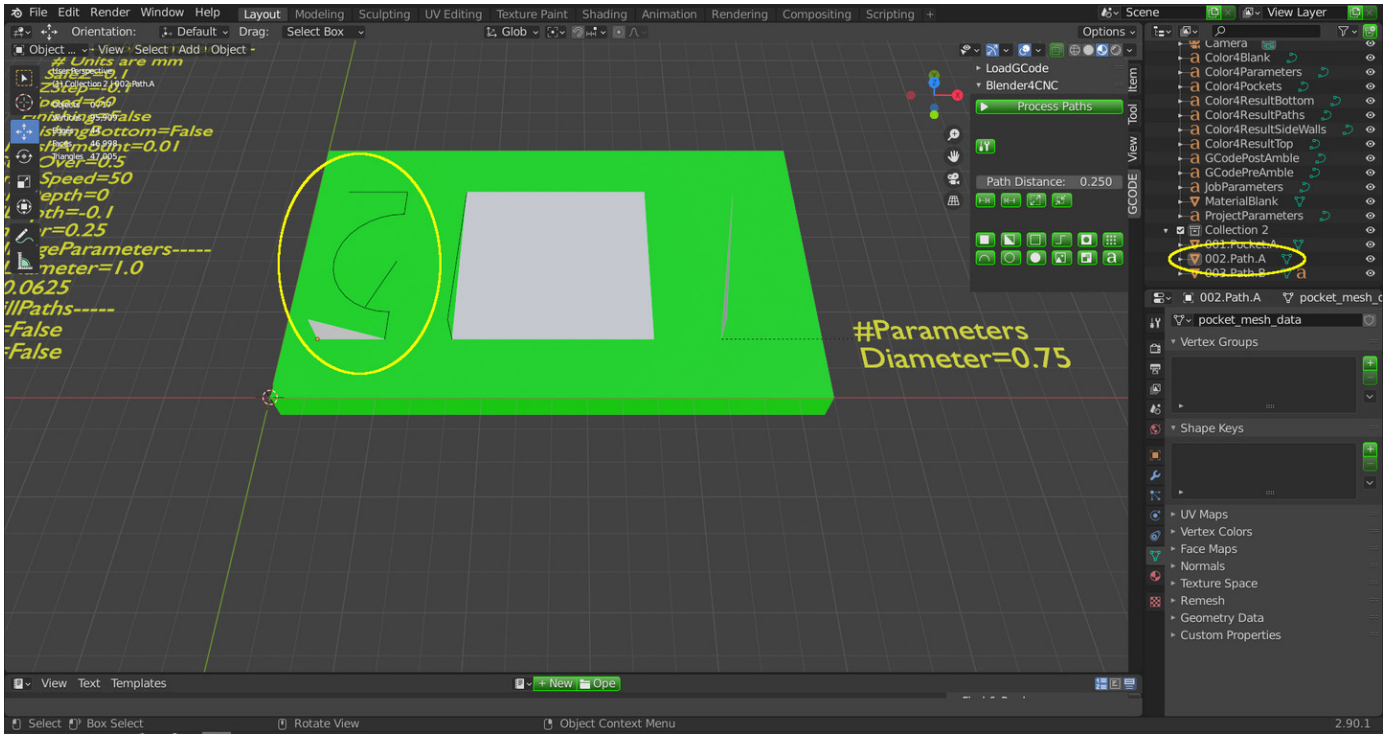

Figure 31: The second operation is a path called 002.Path.A. A path tells the CNC machine to simply follow the path exactly and remove material as it goes.

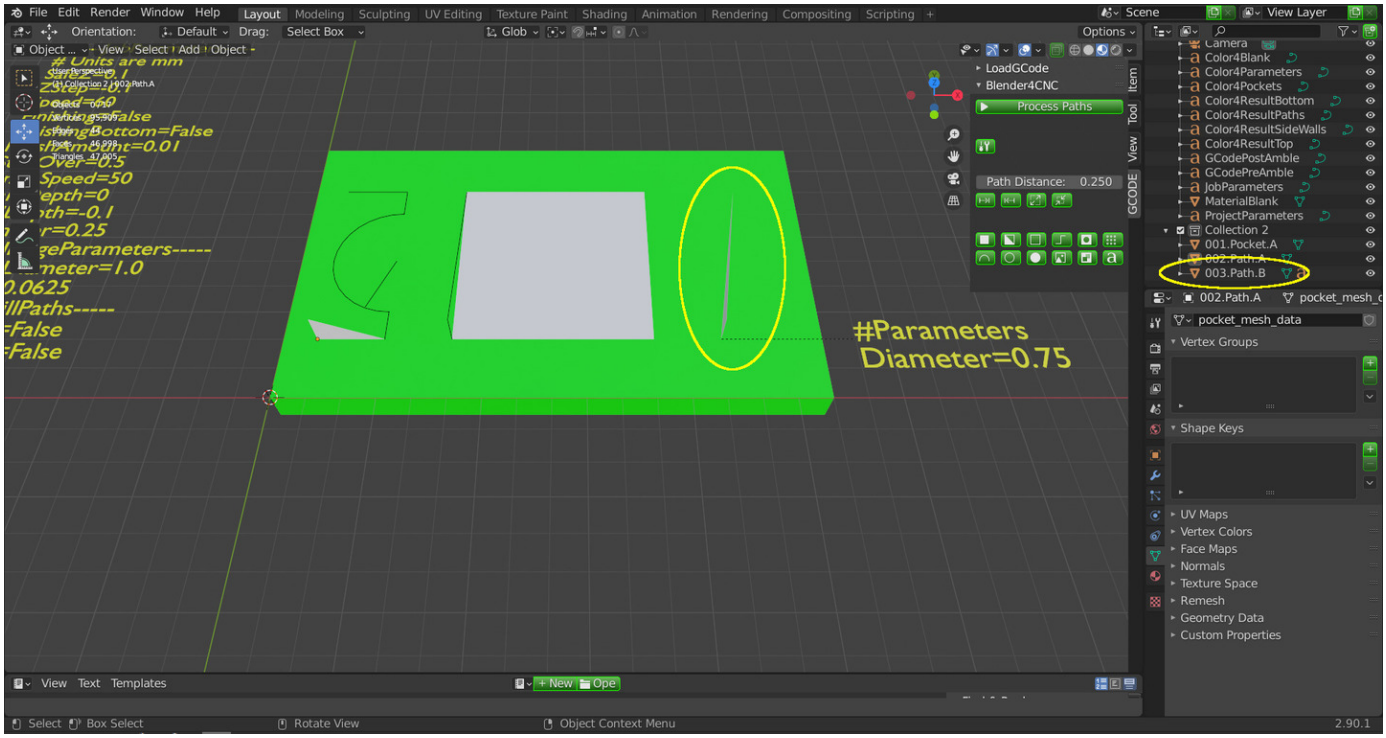

Figure 32: The 3rd operation is also a path (called 003.Path.B) but it is to be cut with a different diameter router bit.

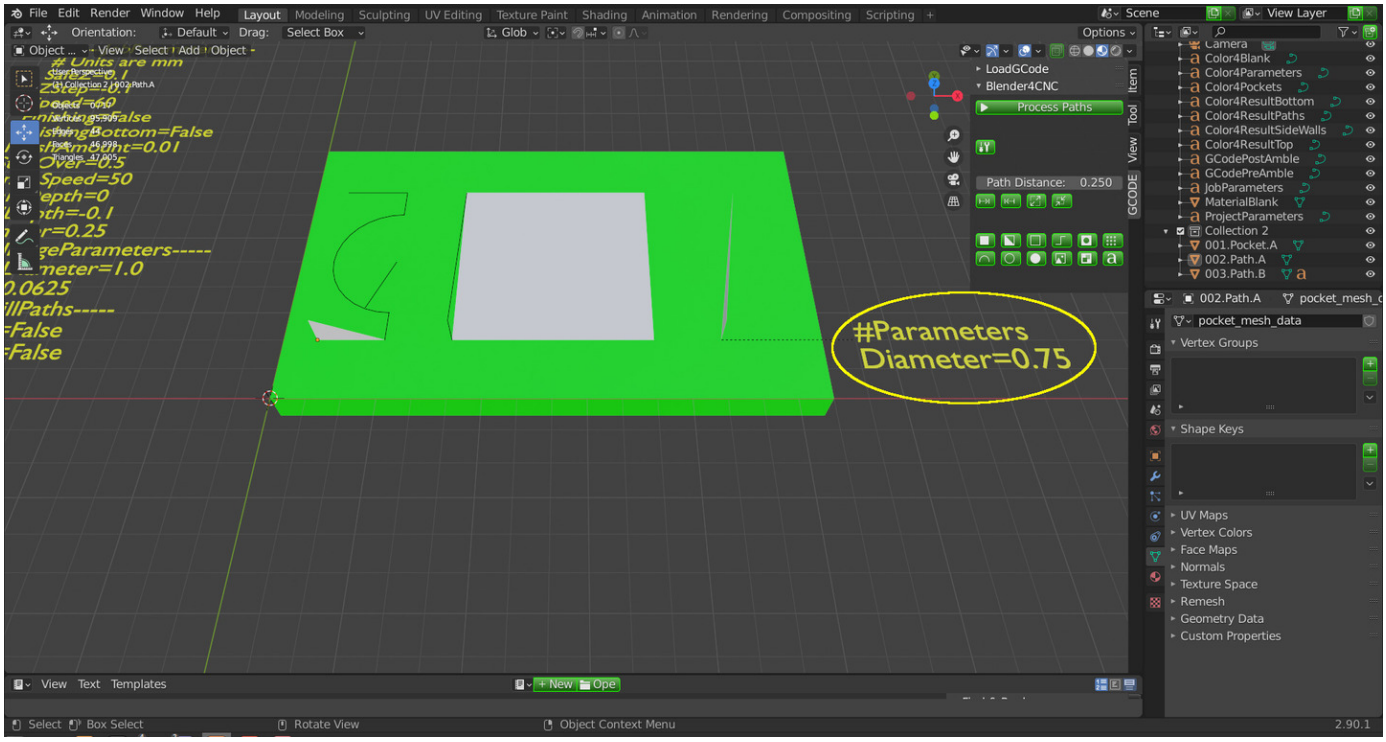

Figure 33: Overriding job parameters for a path: the default job parameters define a router bit diameter of 0.25", while the 3rd operation has a parameter attached to it which specifies this is to be cut at 0.5" diameter. (This requires a tool change on the machine.)

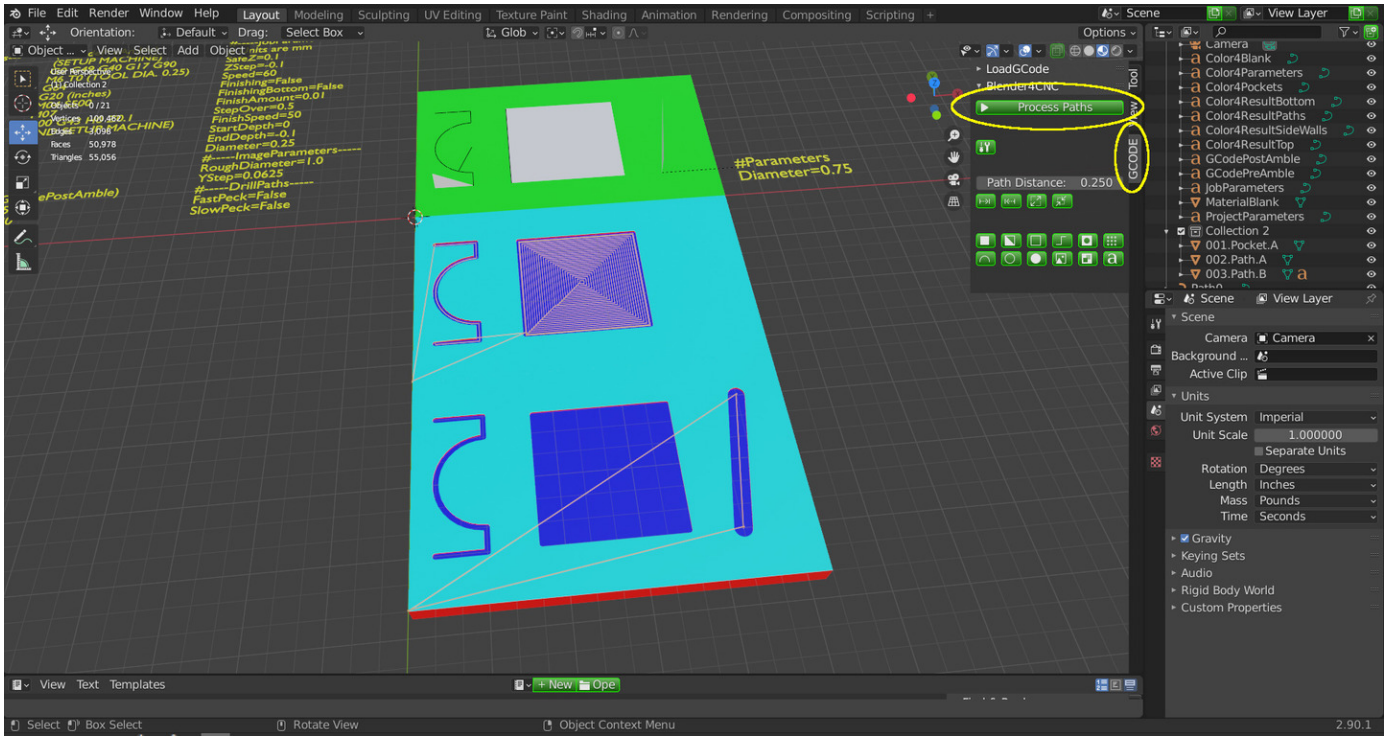

Figure 34: To see the result of these operations, click the "GCode" Tab and then click "Process Paths". Here we see the material as it would look after each set of operations. In this case, the first set of operations were cut at 0.25" diameter and the second set of operations were cut at 0.5". It also shows the exact path the tip of the router bit will follow for each operation in each set.

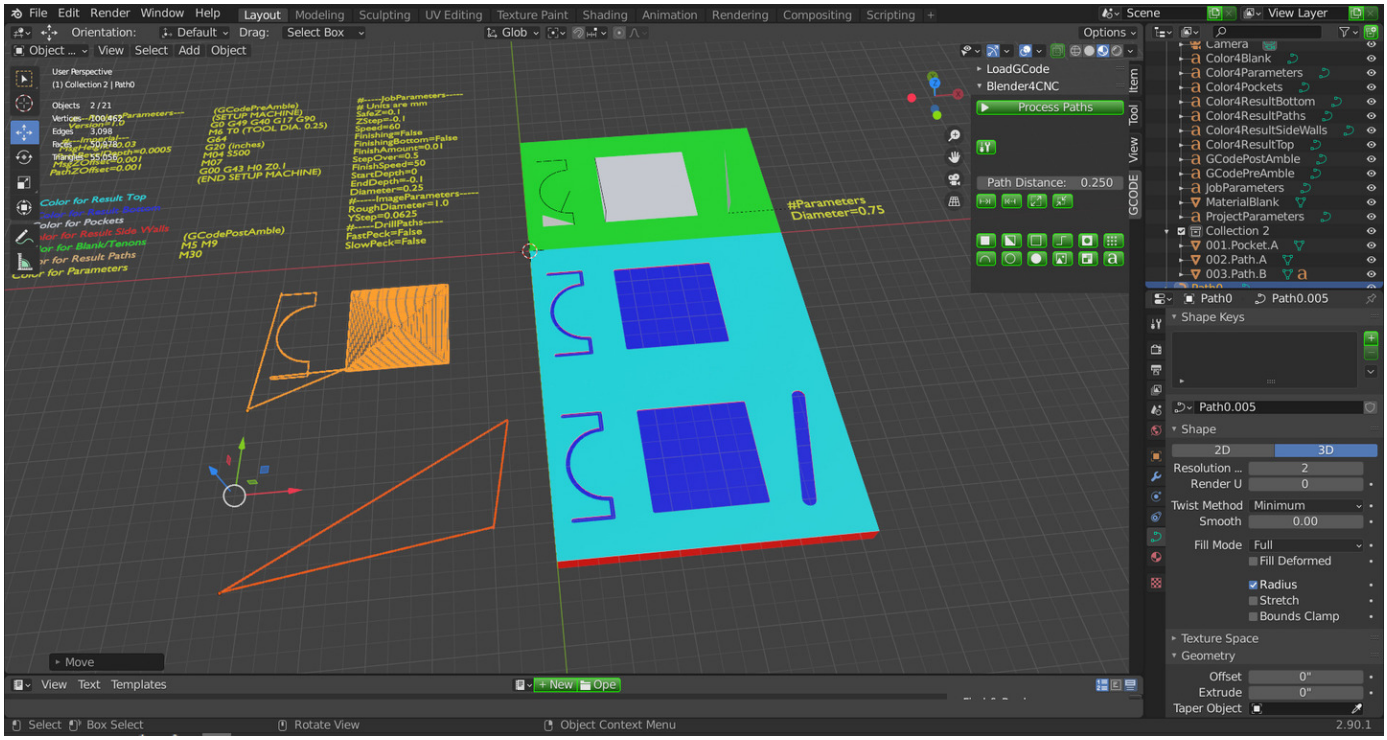

Figure 35: Visualizing the paths - separated for clarity. On the left, are two Blender curve objects which show the path of the router bit. There are two of these because the job required operations with a 0.25" bit; a tool change to 0.5"; then a final operation at 0.5".

When "Process Paths" is clicked, the operations are visualized and G-Code files are produced. These G-Code files, denoted with suffix .ngc, are ready to be sent to the CNC machine. The G-Code files are created in the same folder as the Blender project (the ".blend" file) and are called "{Project Name}-X.ngc" where {Project Name} will match the name of the Blender project and "X" will be a number representing the set of operations. In this case, because the project is called "Addons-Doc.blend", it produced two files called "Addons-Doc-1.ngc" and "Addons-Doc-2.ngc". The first GCode file runs the operations with a router bit of diameter 0.25". The second GCode file runs the operation that uses the 0.5" diameter bit. (The only exception to the output naming convention of {Project-Name} is if the user has specified a parameter "OutputFilename={name}" in the JobParameters in which case the G-Code files are called {name}-X.ngc.)

The "Just Produce GCode" button will only create the GCode files and will NOT visualize the operations. On a complex design this can save some time if you are sure the operations are what you want. (This can also be helpful in the event that a bug occurs in the visualization portion of the software and crashes while trying to render the visualization - you may be able to work around the issue by just generating GCode.)

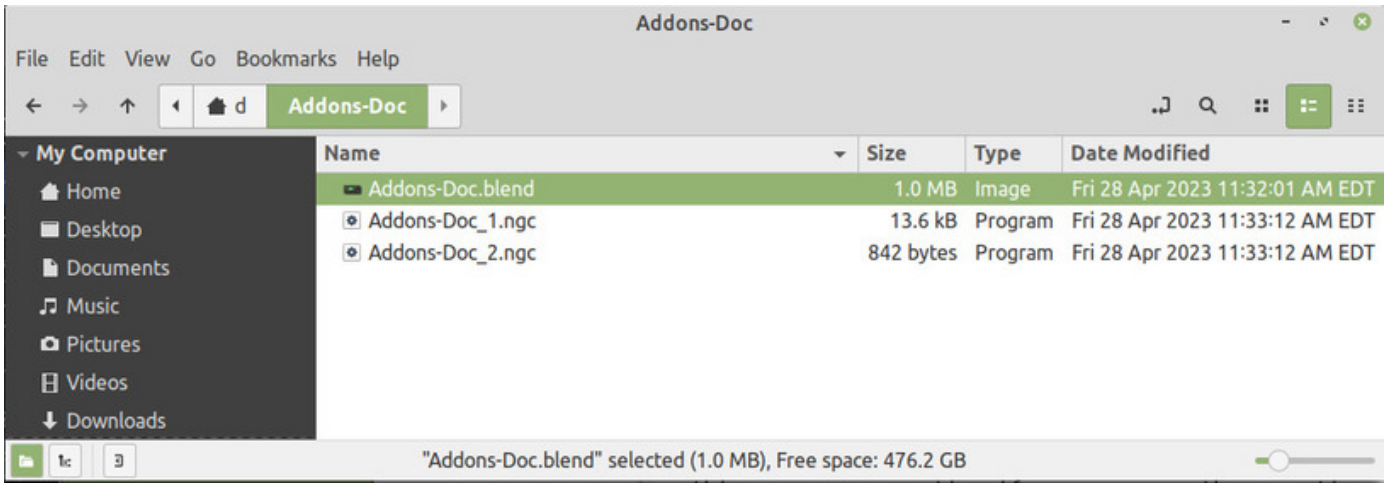

Figure 36: The GCode files are in the folder with the Blender project after clicking on "Process Paths".

1.3 Quickly Create Blender4CNC Operations

All CNC operations in a Blender4CNC project can be created from basic Blender objects. However, Blender4CNC provides utility functions to create operations more quickly. Blender4CNC also provides a template CNC project which serves as an easy way to begin a Blender4CNC project. (Note that the default Blender color for buttons on the screen is **gray** images in this documentation contain a theme where the button color has been changed to green. Images in this documentation have been rotated or zoomed in the user interface - if you are not familiar with Blender you will have to practise zooming and rotating by clicking the middle mouse button and moving the mouse (to rotate) or scroll the middle button (zoom).)

1.3.1 The Project Template.

When you downloaded Blender4CNC, there is a template project named "Template-Imperial.blend" - load it by going to "File" -> "Open". You may need to scroll the middle mouse button to see all the objects in the scene; don't worry if your view doesn't look **exactly** the same as figure 37. There are text objects defining the GCode PreAmble and PostAmble which you will change as necessary for your CNC machine. There are text objects defining the Project Parameters and the Job Parameters. There is a 16"x10"x1" material blank. In the top right window, there is an empty collection named "Collection 2". In the lower right, this project is set to "Imperial" units. The units of measure when working in Imperial is "Inches". (Blender4CNC also contains a template project for metric users named "Template-Metric.blend". If you would prefer to use the metric template throughout this section feel free to do so but you will have to adjust some of the instructions to place example operations at say 20mm instead of 1inch.)

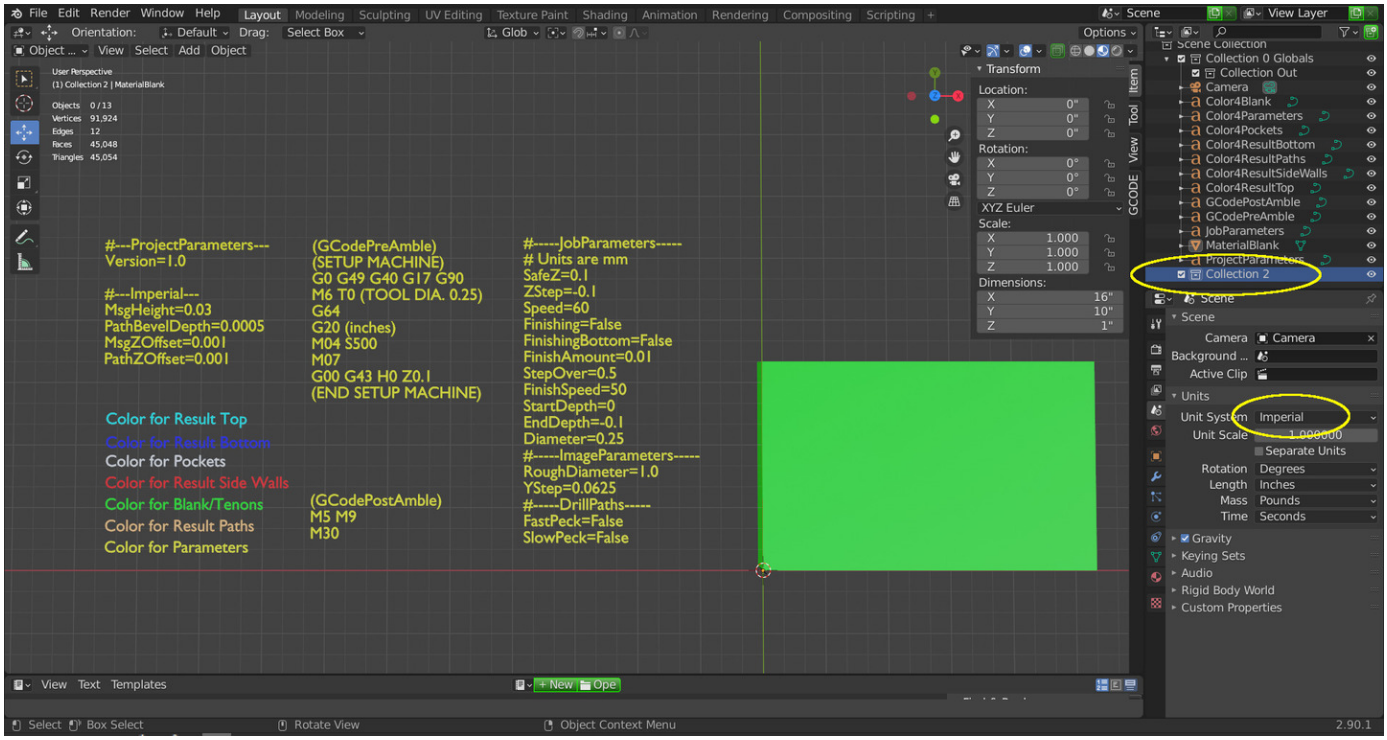

Figure 37: The template project.

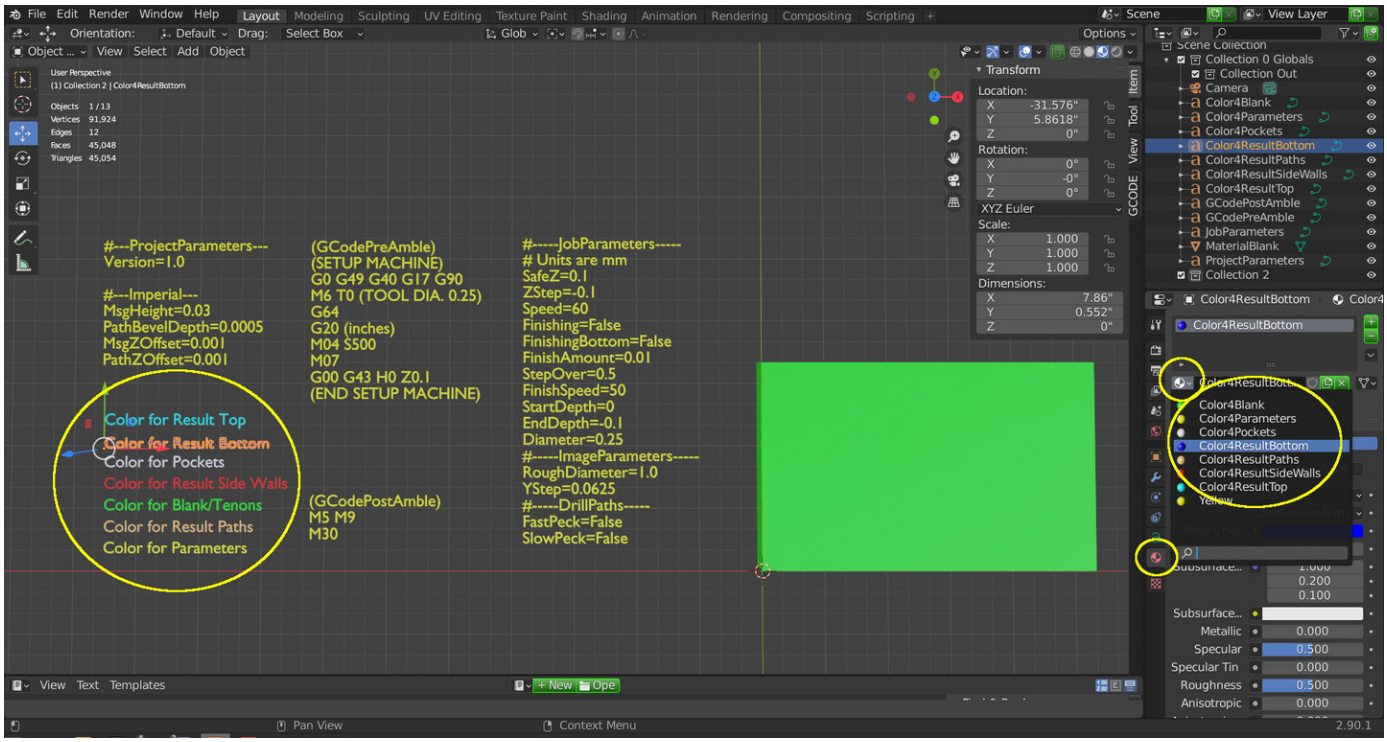

Figure 38: There are also seven special materials (or colors) defined in the template. These colors are used when you create Pockets, Parameters, Tenons and when the results are visulalized. You can change these color materials to your preferences. When you create any type of Pocket it will get assigned the "Color4Pockets" material (which in my case is gray). Likewise, Parameters will be "Color4Parameters" (yellow); Tenons will be "Color4Blank" (green - the same as the 10x10x1 blank); output results will be shown in the colors "Color4ResultTop", "Color4ResultSideWalls" and "Color4ResultBottom"; output paths (showing the movement of the router bit) will be "Color4ResultPaths". To change any of these colors, you can select the relevant "Color for ..." text object in the upper right corner and change the color of the material (lower right corner) associated with it.

1.3.2 Quickly Create Operations (using a Template) - Pockets, Paths, Holes, Circles and Arcs.

The PreAmble is GCode that is placed at the beginning of every GCode file; the PostAmble is GCode that is placed at the end of every GCode file. (In Blender, you change text objects by selecting the text object with the left mouse button and pressing TAB - this lets you edit the text - press TAB when finished.) Once you have changed the PreAmble and PostAmble you should save the template to a new template name that you can recall for future projects. Go to "File", "Save As" and save the project with a new name (so you don't accidently overwrite the template file). Blender4CNC is not (yet) knowledgeable about the specifics of other common CNC machines so it does not offer "profiles" to be loaded for each type or brand of CNC machine. There are text objects defining some default parameters for the Project and the Job. (You probably do not need to change the Project Parameters.) In the top right window, there is an empty collection named "Collection 2". **Click on "Collection 2" to make sure it is selected.**

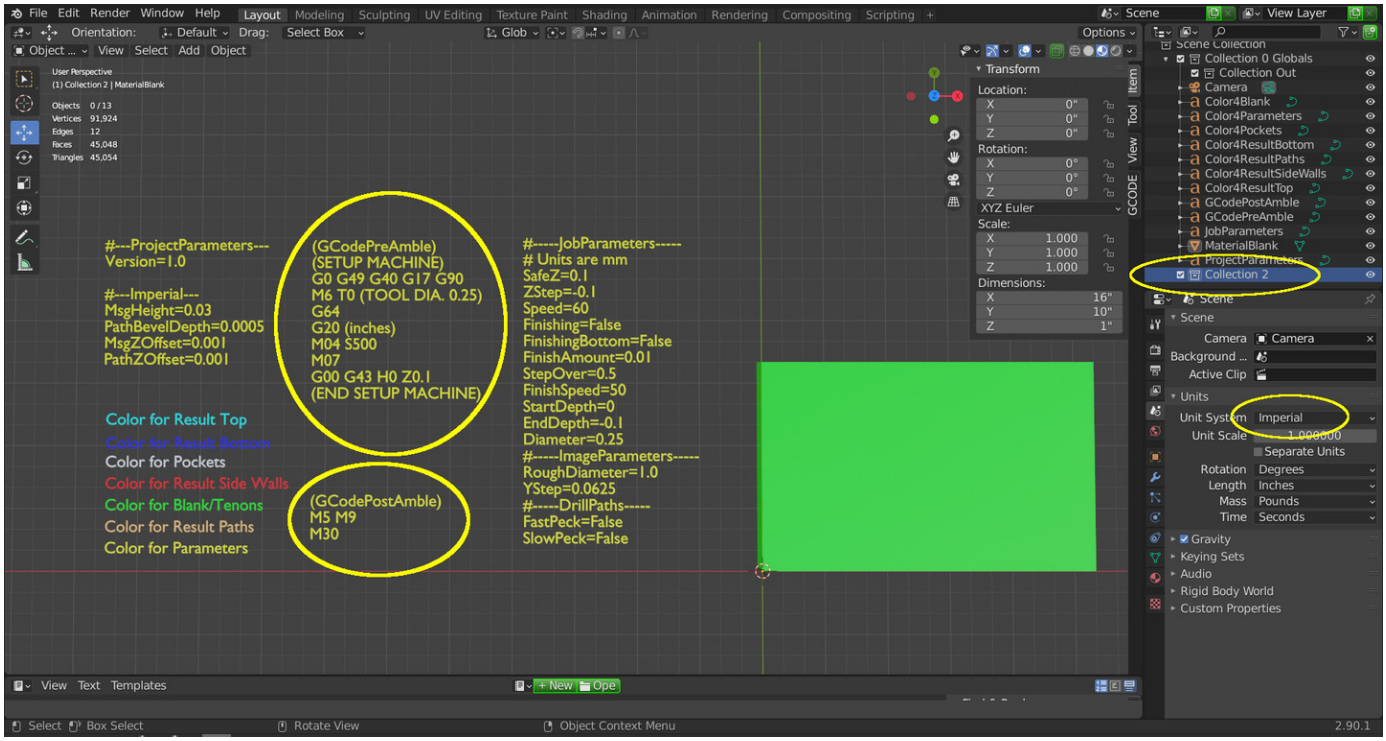

Figure 39: PreAmble and PostAmble.

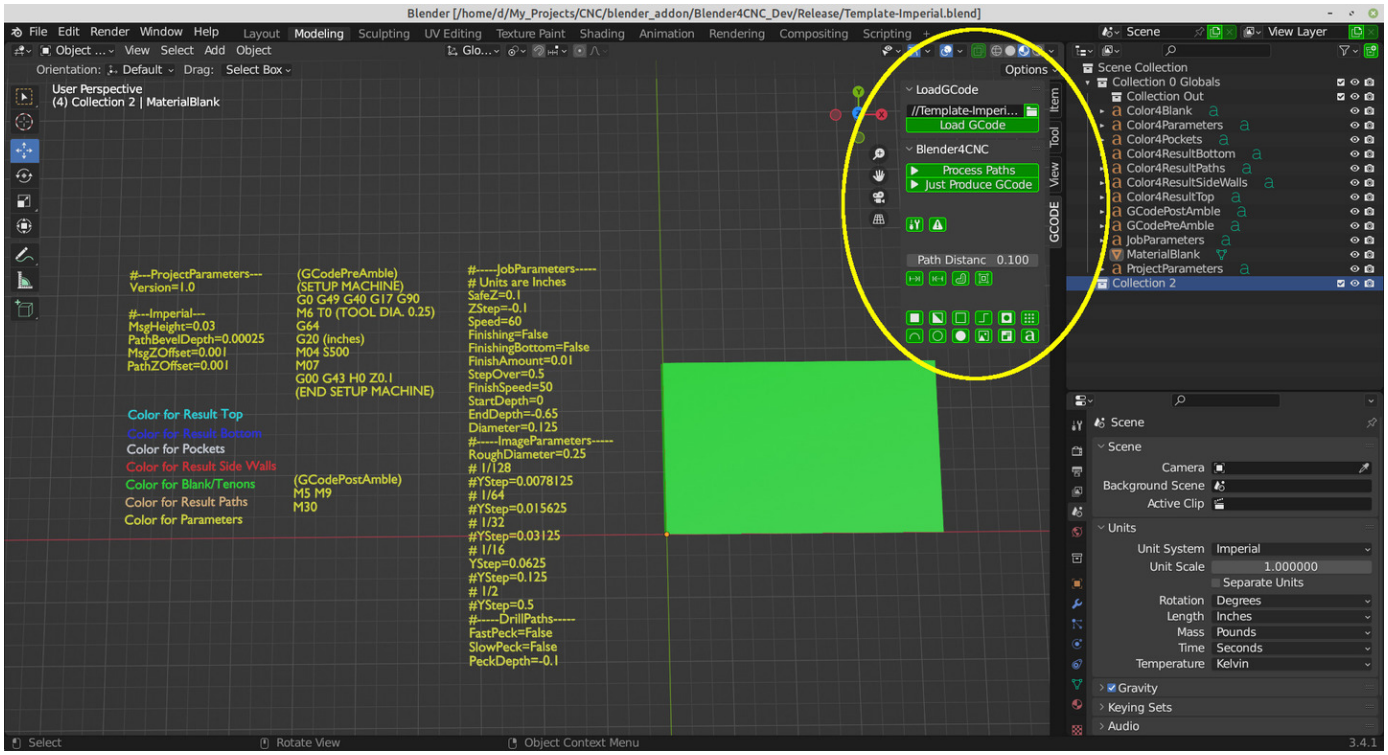

Figure 40: Blender4CNC functions are located on the right-hand side of the 3D View workspace under the "GCODE" tab along with the "Item", "Tool", and "View" tabs . If you do not see these tabs, see the following figure. If you do see these tabs, click the GCODE tab to see the Blender4CNC functions and buttons.

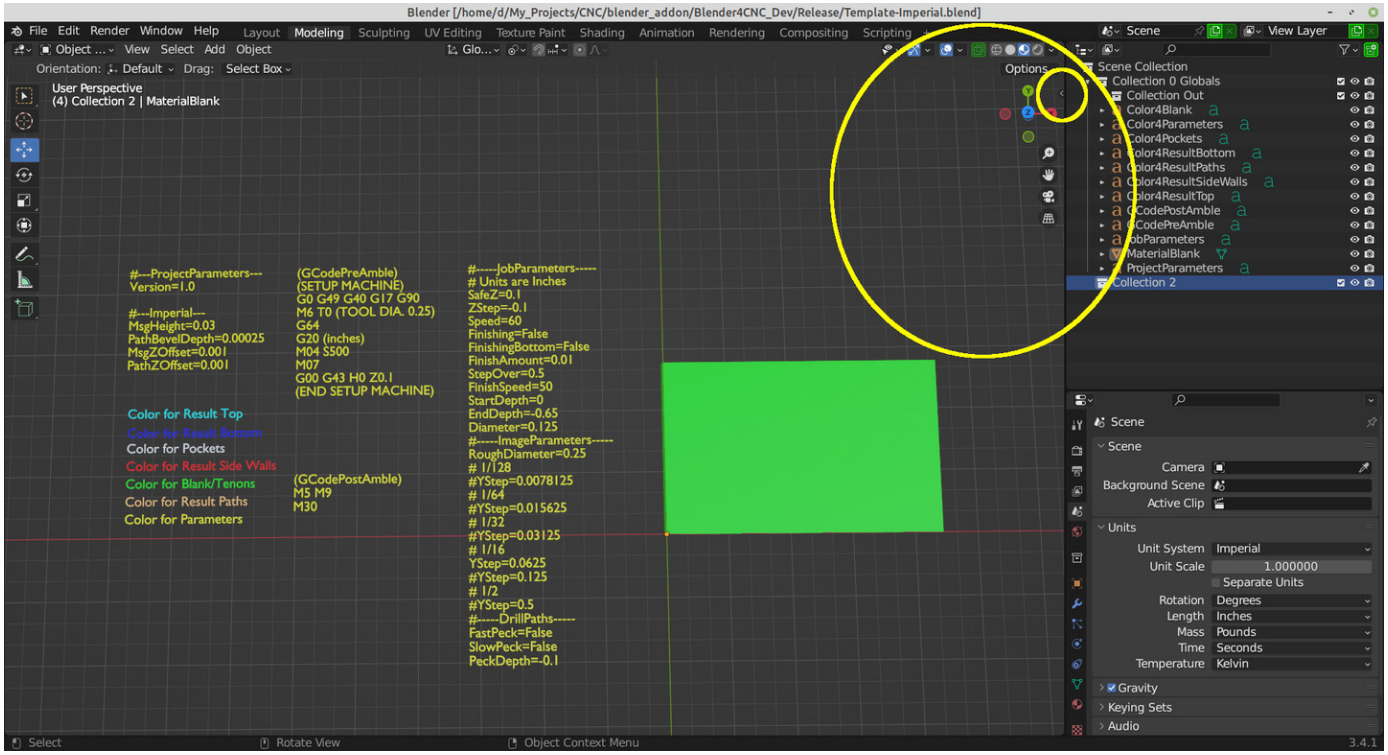

Figure 41: When Blender is first installed, it sometimes defaults to hiding the "Item", "Tool", and "View" tabs. Note there is a tiny arrow that looks like "<". Click this arrow and they should appear. Click the GCODE tab to see all the Blender4CNC functions and buttons.

Cut CW Pocket (clockwise pocket)

Cut CCW Pocket (counter-clockwise pocket)

Cut Closed Path

Cut Open Path

Drill Hole

Drill many Holes

Create an Arc

Cut a Circle Path

Cut a Circle Pocket

Cut a Depth Image

Create a Tenon (an island within a pocket)

Create a Parameter

Figure 42: The buttons for creating CNC operations.

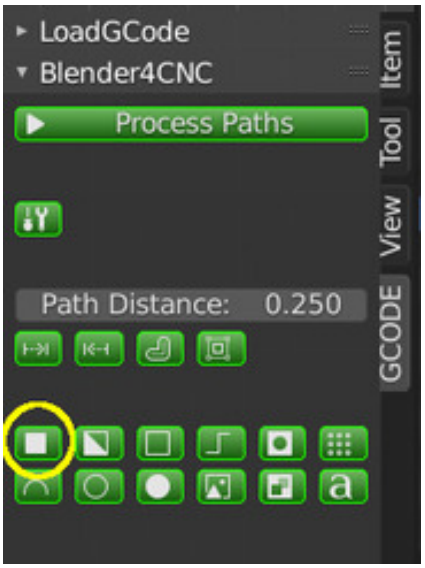

Figure 43: **Make sure that "Collection 2" is selected so that the following operations are correctly added into Collection 2".** Click the "Create CW Pocket" button and you should see a pocket operation appear on the lower left corner of the material blank.

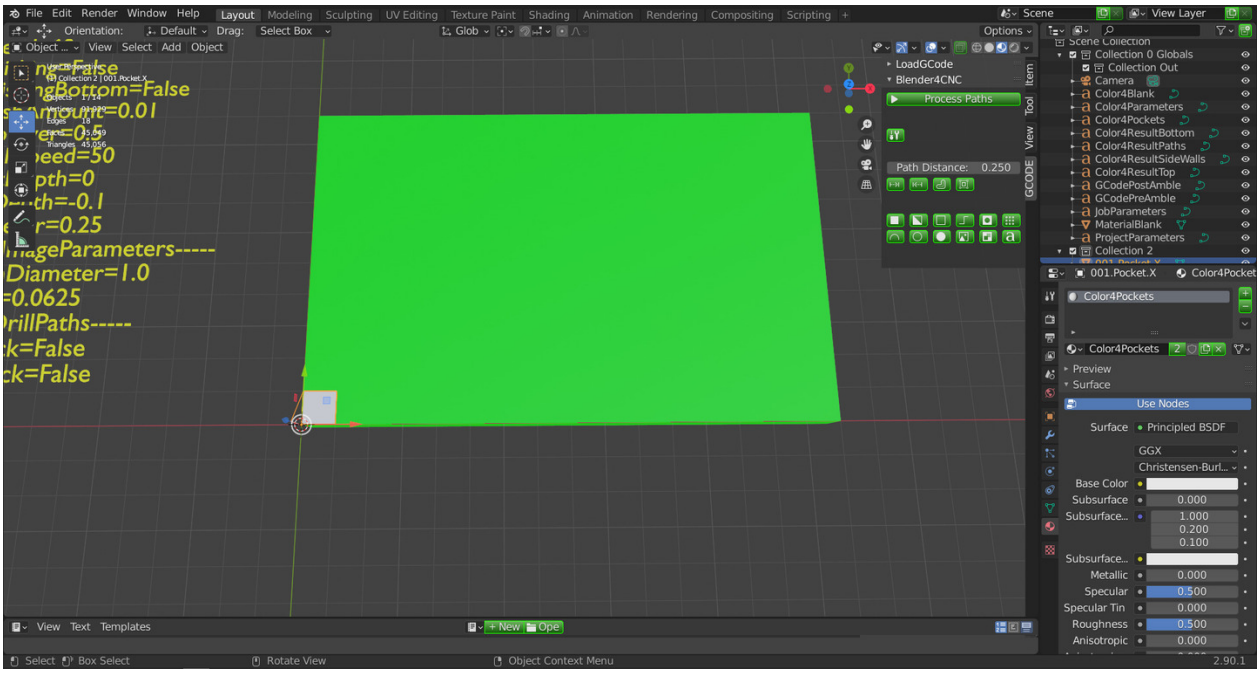

Figure 44: A pocket operation is added.

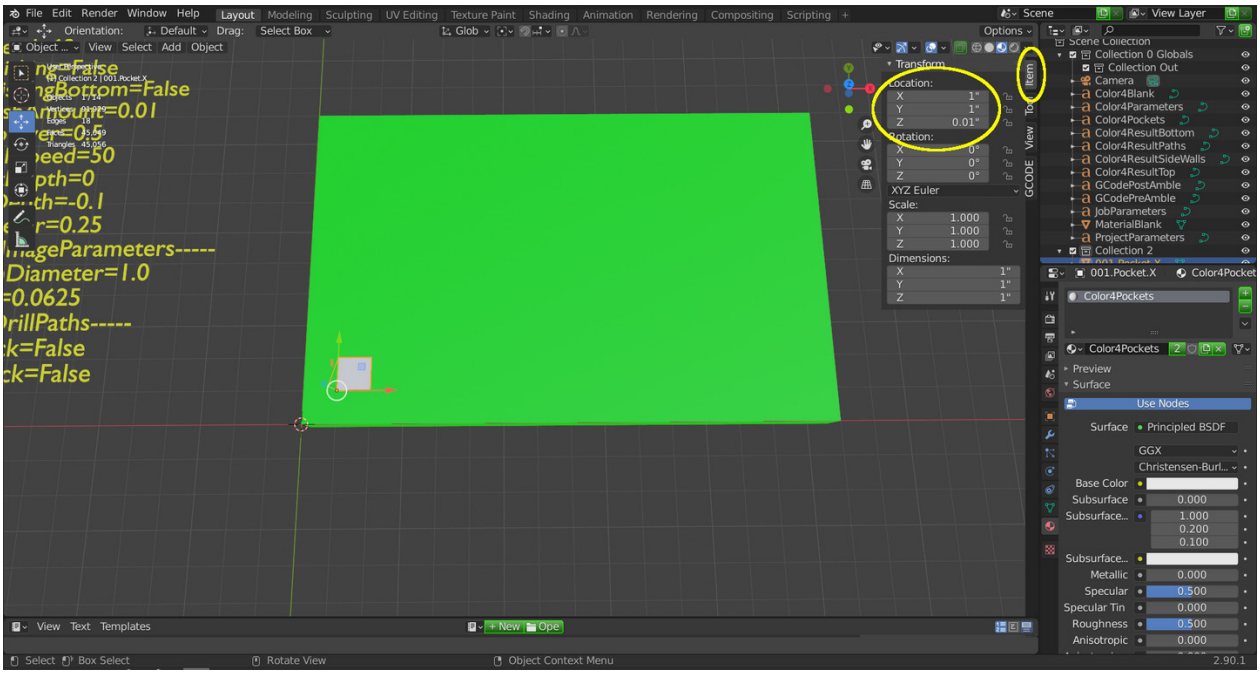

Figure 45: Move the pocket over by clicking on the "Item" tab and then editing the X and Y location fields to 1.0 and 1.0.

Let's generate GCode for this simple pocket and visualize the result. Under the GCODE tab, click on "Process Paths".

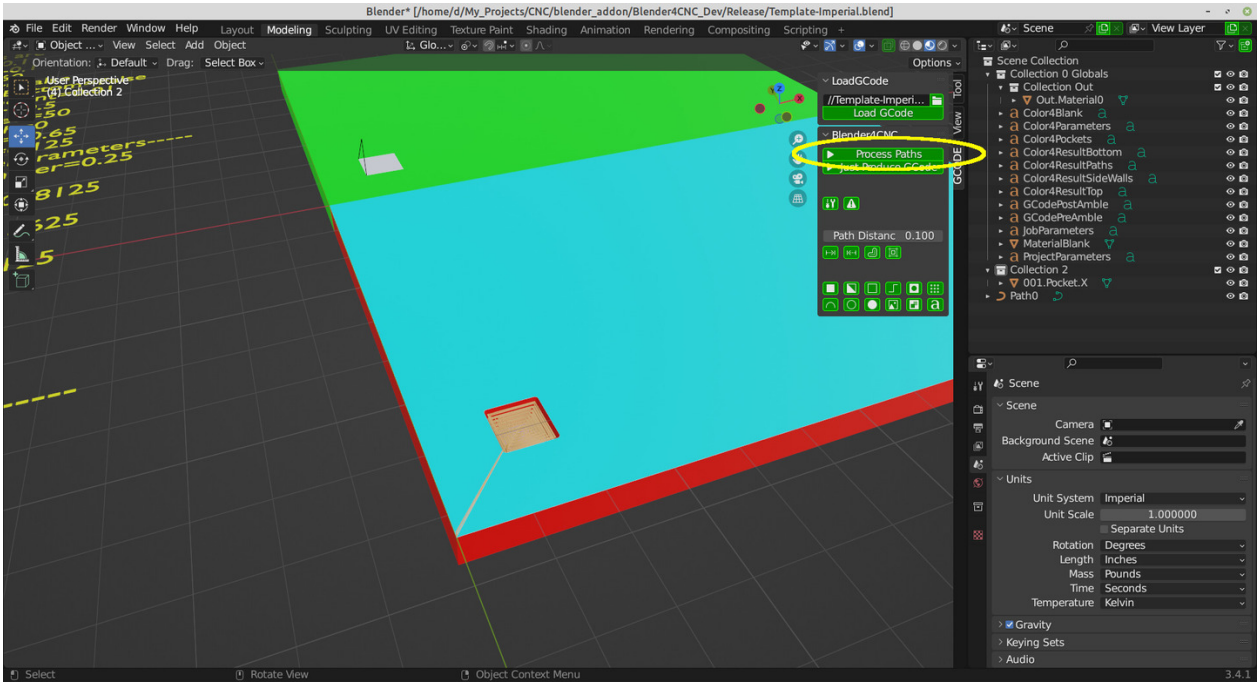

Figure 46: A simple square pocket is visualized and a text file containing the GCode has been written to disk.

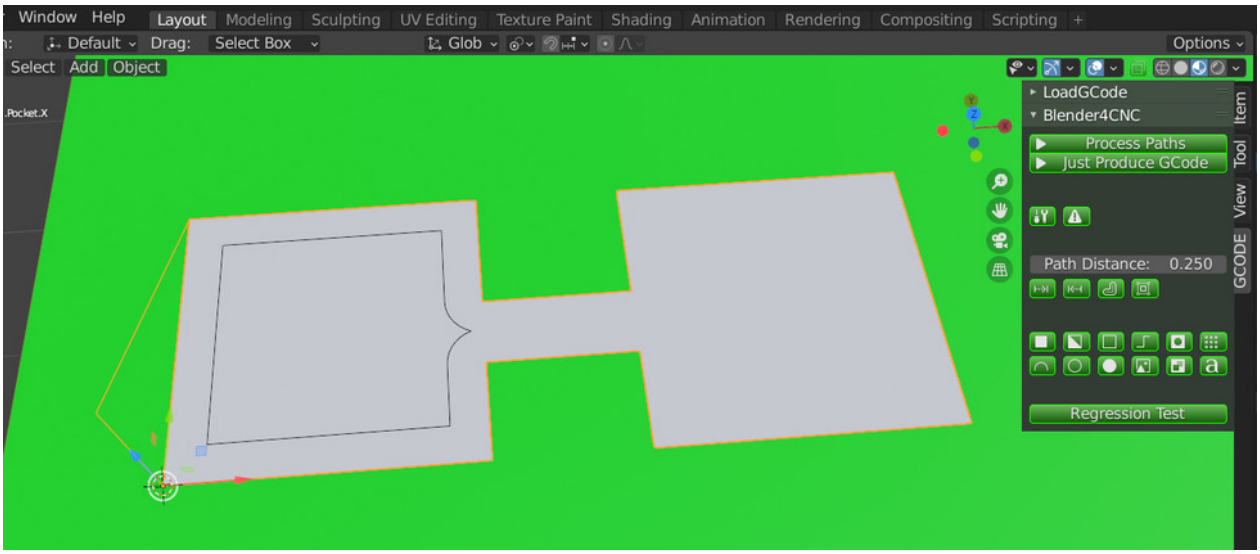

Figure 47: (A note for future reference when creating more advanced shapes and pockets: if a pocket contains an area which the cutter cannot reach because the cutter is too large to fit between two vertices in the shape, an error will occur.)

Now click on the "GCODE" tab.

Figure 48: Click the "Create CCW Pocket" button and a pocket operation (climbing cut) will appear.

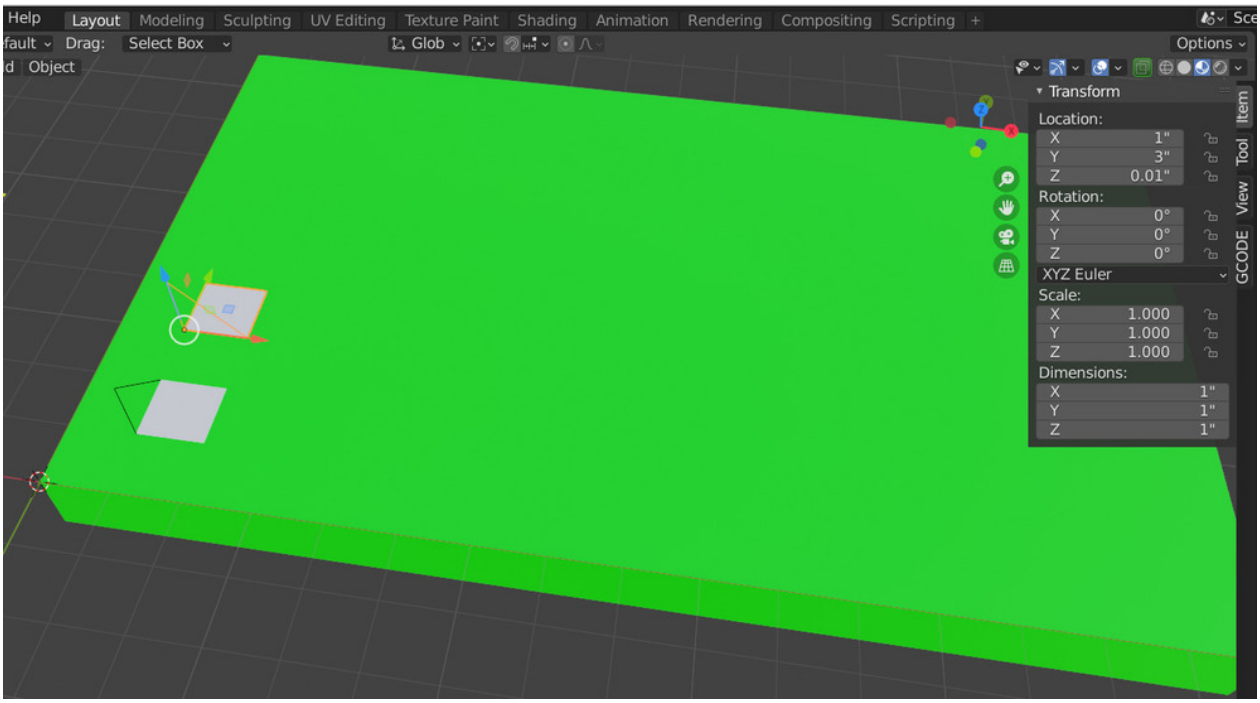

Figure 49: A counter-clockwise pocket operation is added. Use the Item tab to edit the X,Y location to 1,3. You should be able to see that the newly created pocket has a reversed start direction.

Now click on the "GCODE" tab.

Figure 50: Now click the "Create Closed Path" button and a path operation will appear.

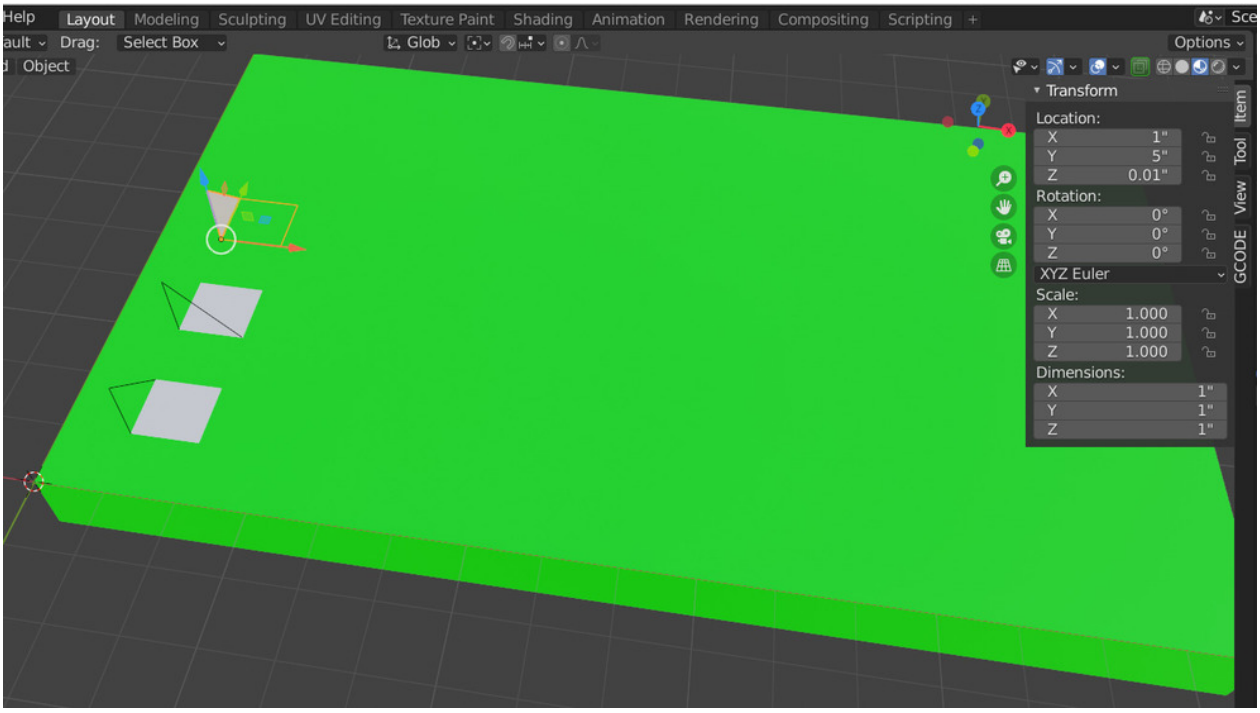

Figure 51: Change the location to 1,5 on the Item tab.

Now click on the "GCODE" tab.

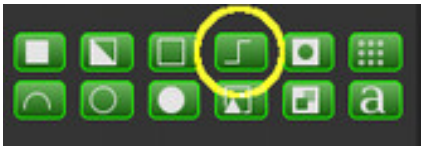

Figure 52: Now click the "Create Open Path" button and a path operation will appear.

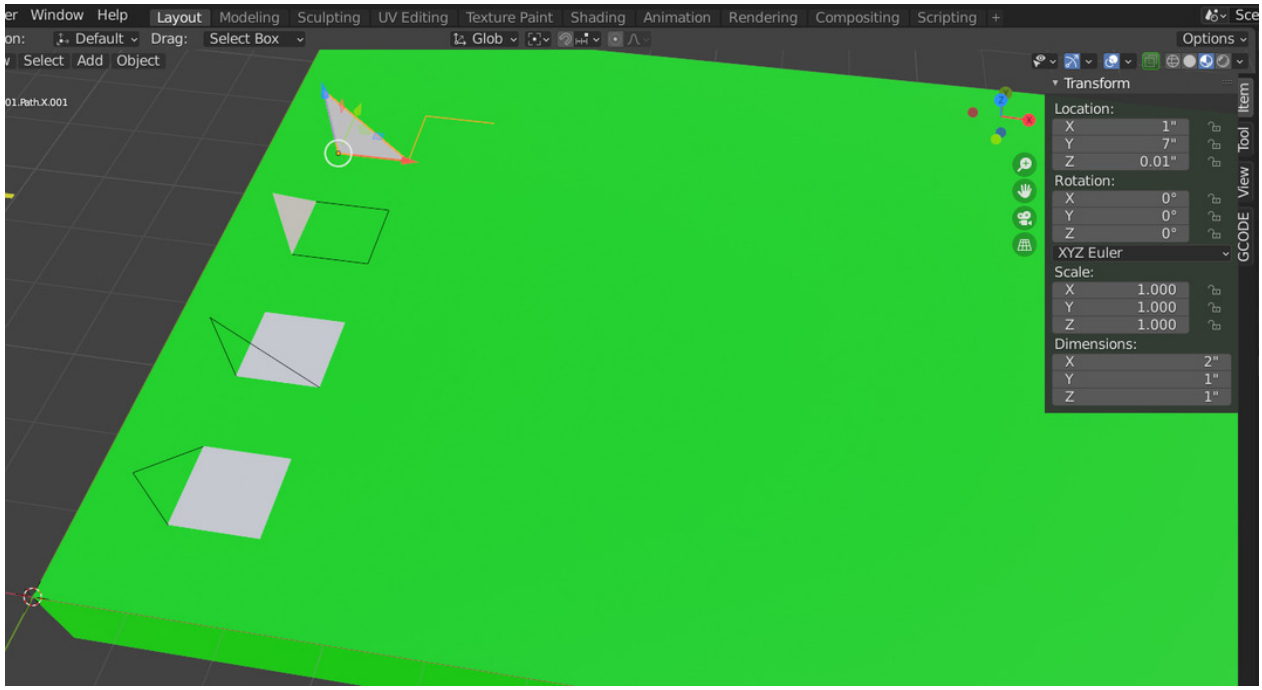

Figure 53: Change the location to 1,7.

Figure 54: Now click the "Create Hole Path" button and a hole operation will appear.

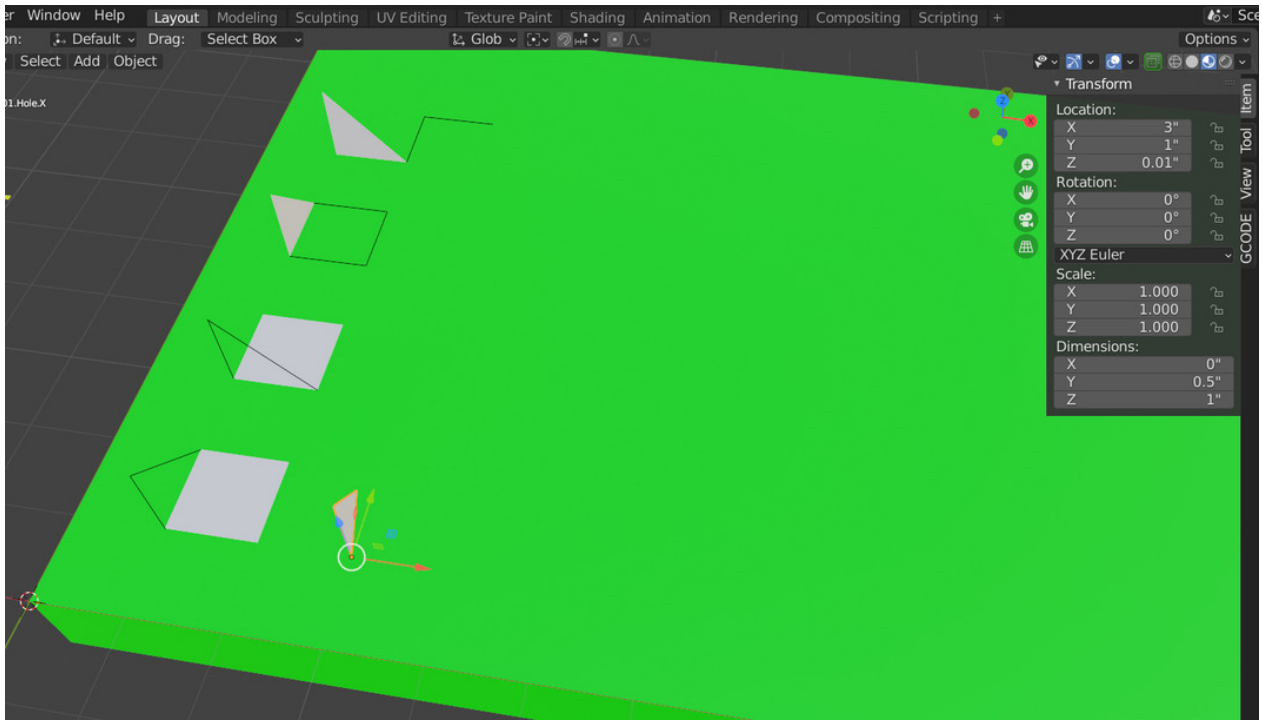

Figure 55: Change the location to 3,1.

Figure 56: Now click the "Create DrillPath" button and a drill path operation will appear.

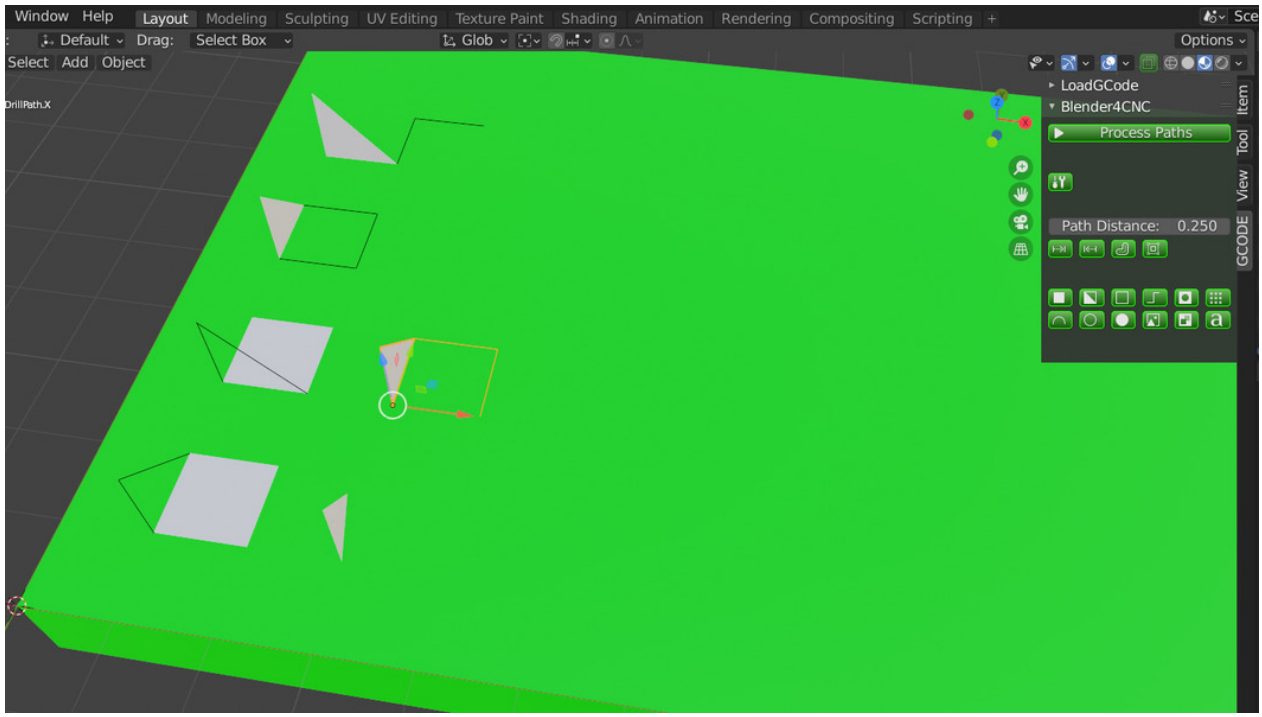

Figure 57: Change the location to 3,3.

Figure 58: Now click the "Create Arc Path" button and an arc path operation will appear.

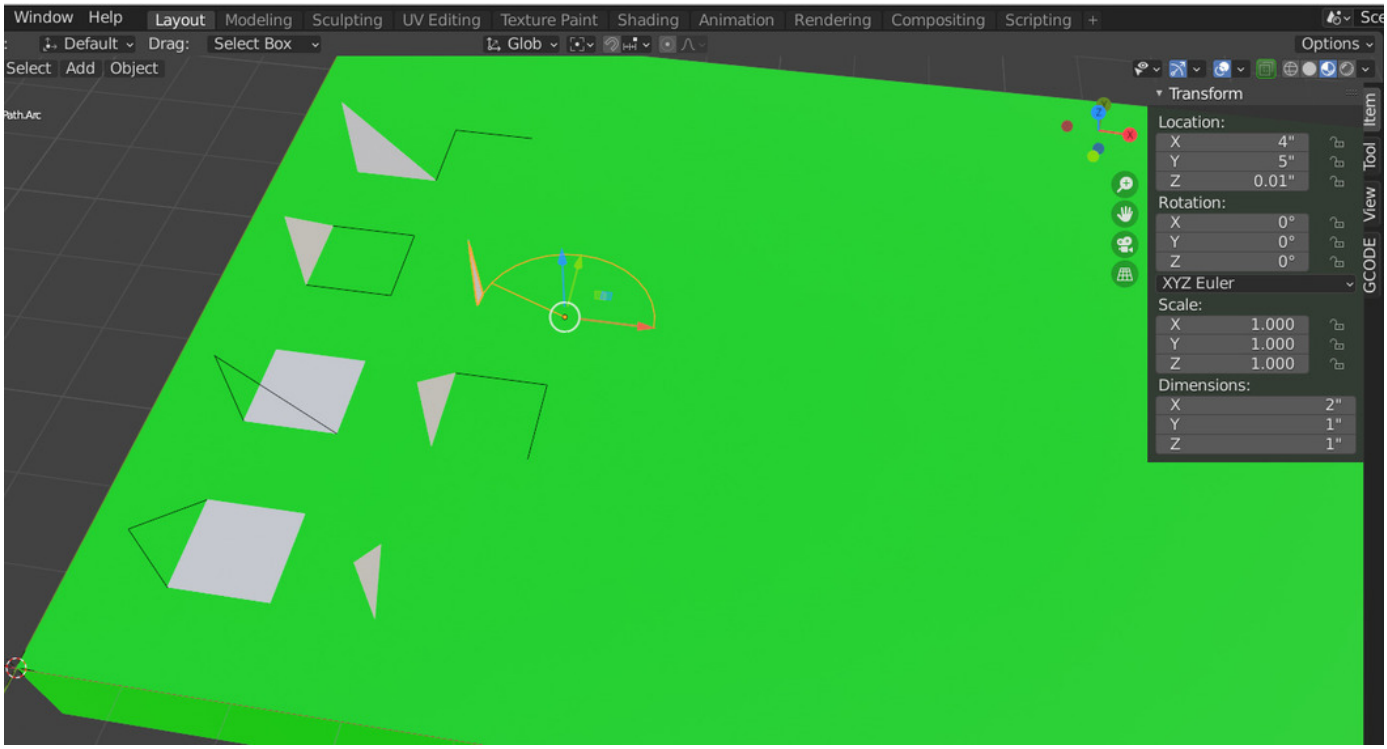

Figure 59: Change the location to 4,5.

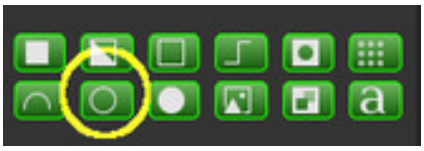

Figure 60: Now click the "Create Circle Path" button and a circle path operation will appear.

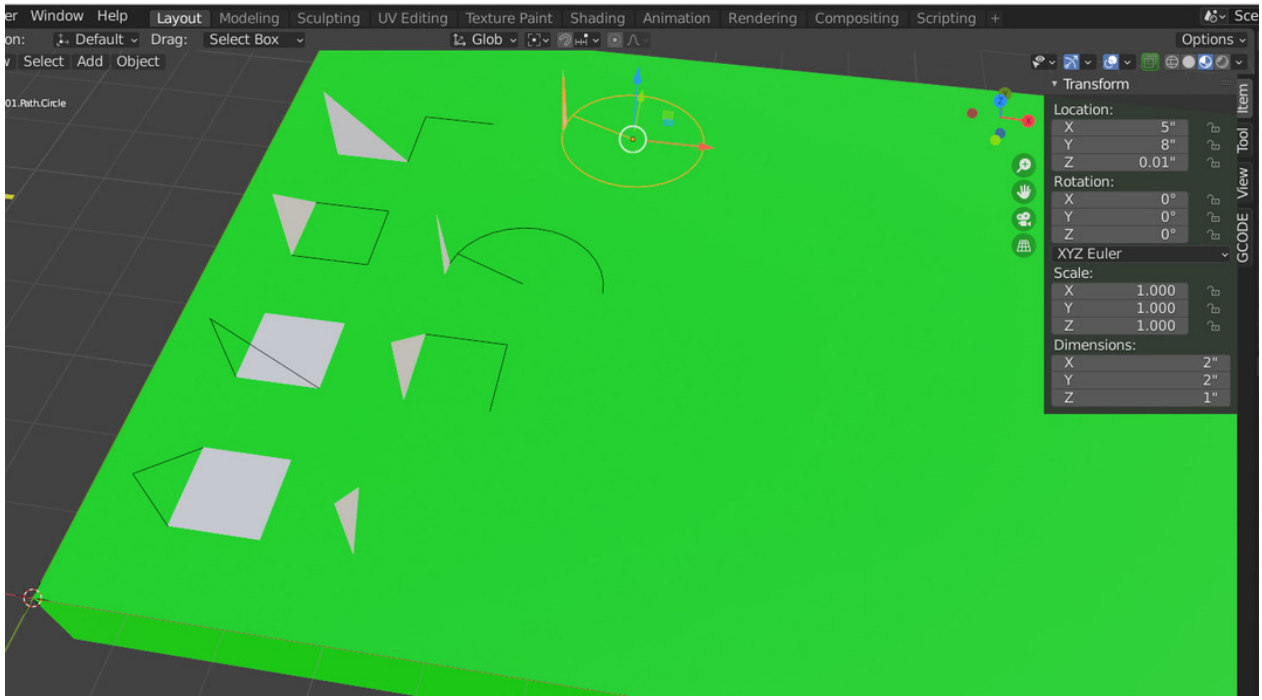

Figure 61: Change the location to 5,8.

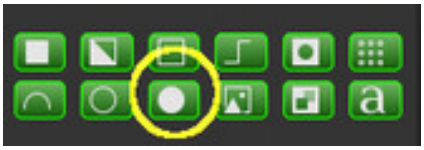

Figure 62: Now click the "Create Circle Pocket" button and a circle pocket operation will appear..

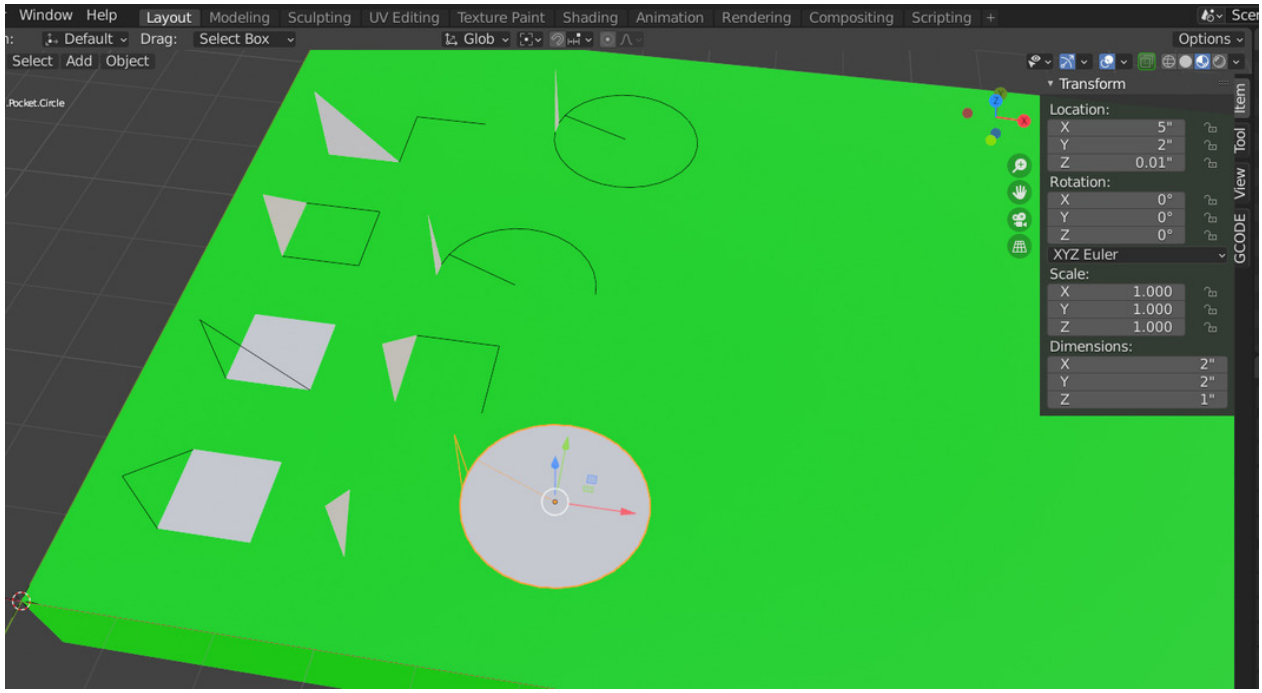

Figure 63: Change the location to 5,2.

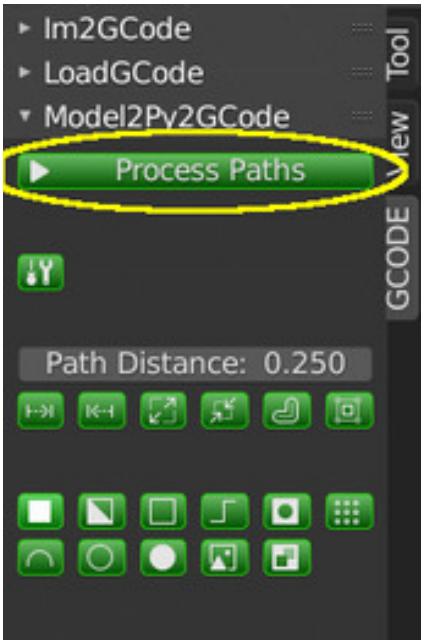

Figure 64: We have created 9 routing operations so far. Let's visualize the operations and produce G-Code. Click the GCODE tab and then click the "Process Paths" button and see the result.

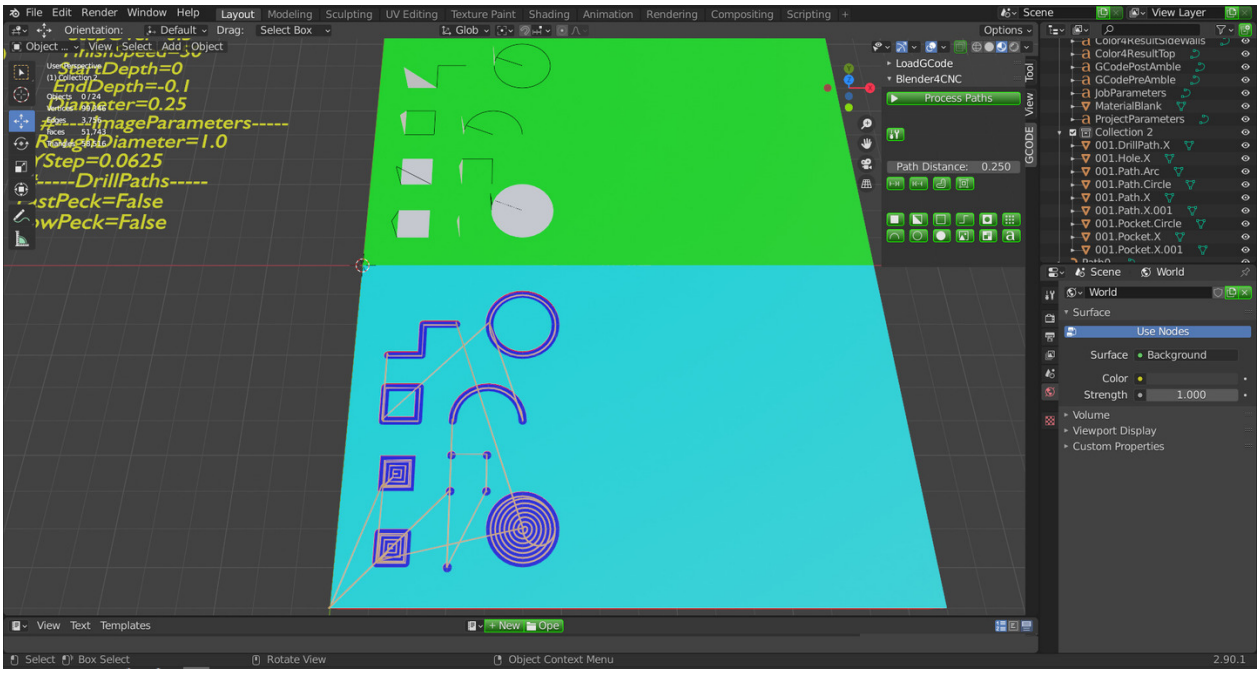

Figure 65: This shows the resultant material in cyan and the router path in red. (If you get an error message, see the next figure.) Note that when generating a path for the cutter, Blender4CNC assumes it starts and ends at (X,Y) = (0,0). However, the generated GCode does not move the cutter to 0,0 at the start or end of the job. On a physical CNC machine, the cutter will end at the "SafeZ" height at the end of the last operation (and it will start from wherever the cutter is currently positioned on the CNC machine when the GCode is run).

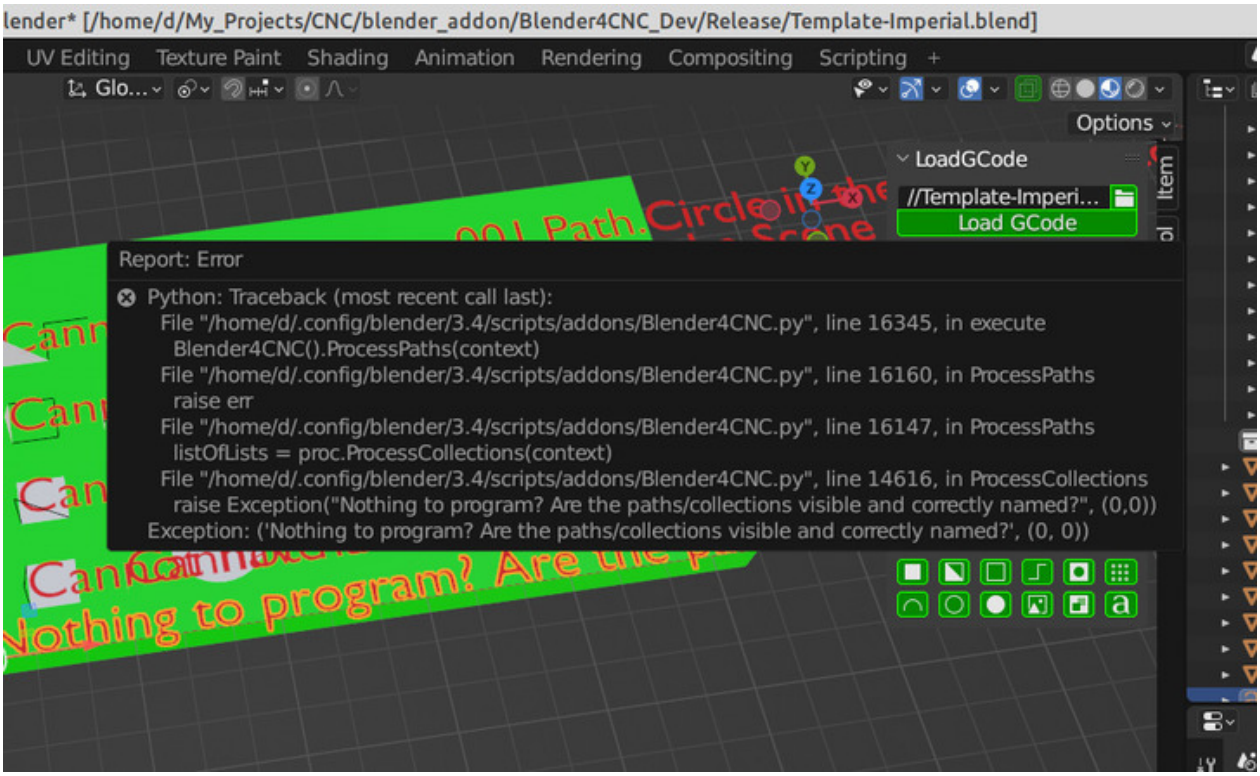

Figure 66: This shows that an error occurred when trying to process the operations. This happened because the operations were not created in "Collection 2". Every operation must be grouped inside a collection.

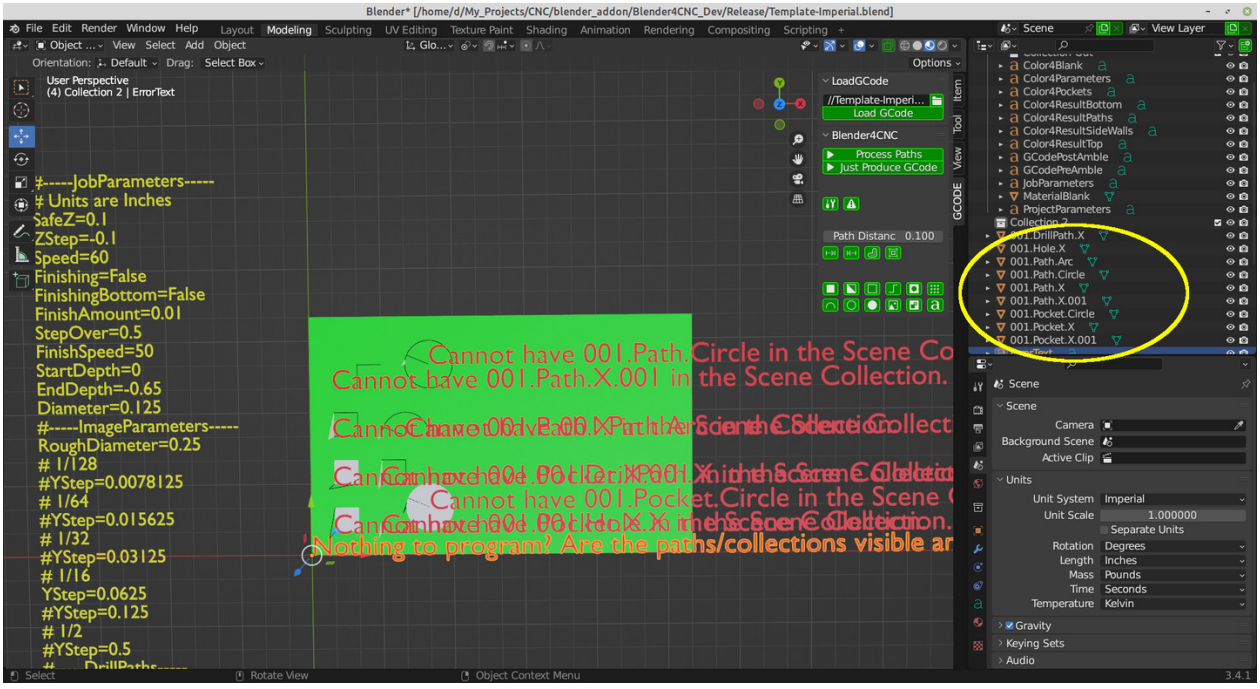

Figure 67: You can see that all the operations on the right side of the screen are not properly in "Collection 2". There is an error message displayed over each operation in the main viewing area in red. To fix this, drag the operations into "Collection 2" with the mouse, then click "Process Paths" again. All the red error messages should go away and you should see the result shown in 65.

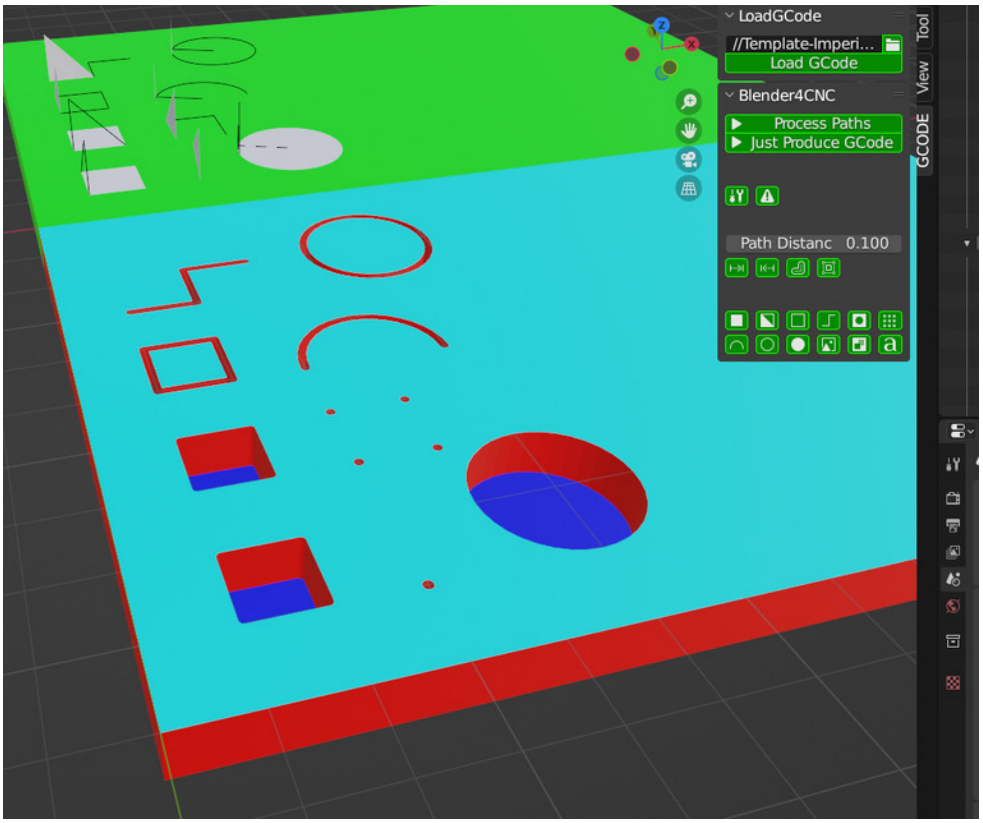

Figure 68: When visualizing the GCode, Blender4CNC creates objects that show the path of the tool and objects that show how the material should look once cut. Here we have deleted the object that shows the tool path (by selecting the yellow path object and hitting "delete") and we can see more clearly how the material will look if we cut all these operations.

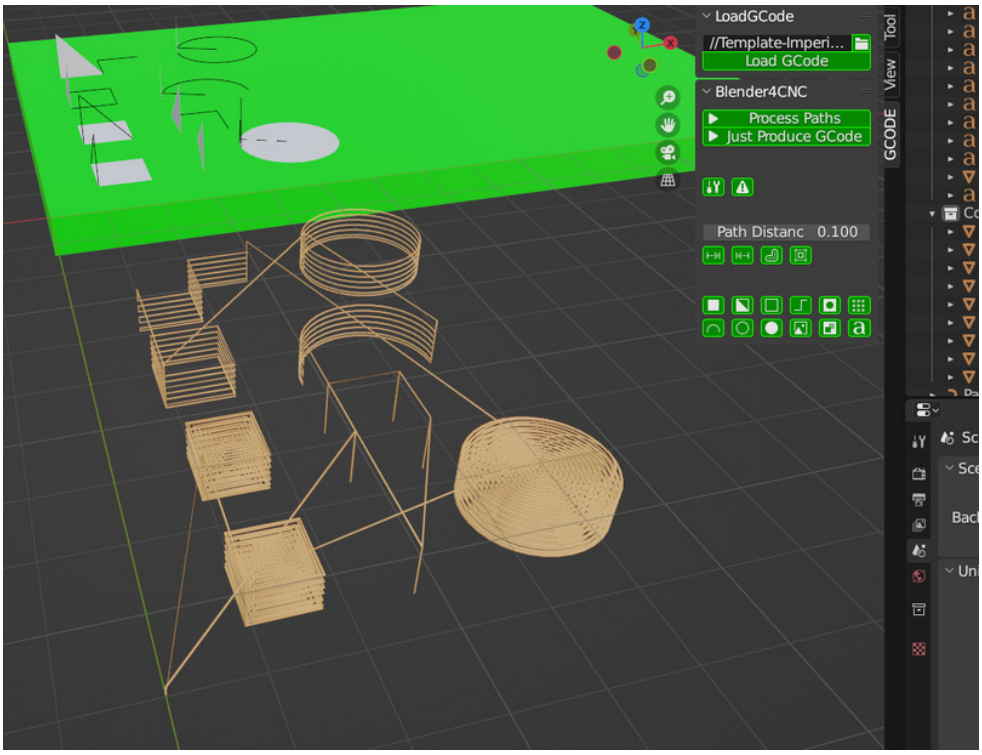

Figure 69: Here we have deleted the object that shows how the material will look (select the cyan object and hit "delete"). We can see more clearly the exact path of the tool and how it successively moves down in each pocket cutting a layer at a time before moving onto the next operation

1.3.3 Quickly Create Operations (using a Template) - A pocket with a Tenon (Island).

From my wood working hobby I learned about mortise and tenon joints (see figure 70) therefore when I want to cut a pocket and leave an "island" of material inside the pocket, I call it a "tenon" within the pocket.

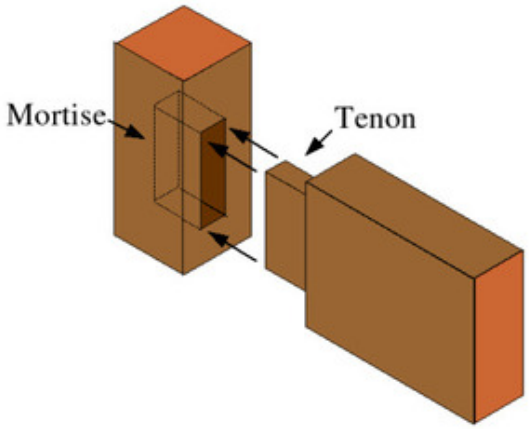

Figure 70: A tenon is material that is left after material is removed around it.

Open the template project named "Template-Imperial.blend". Go to "File", "Save As" and save the project with a new name (so you don't accidently overwrite the template file). **Click on "Collection 2" in the upper right window of the display to make sure it is selected.**

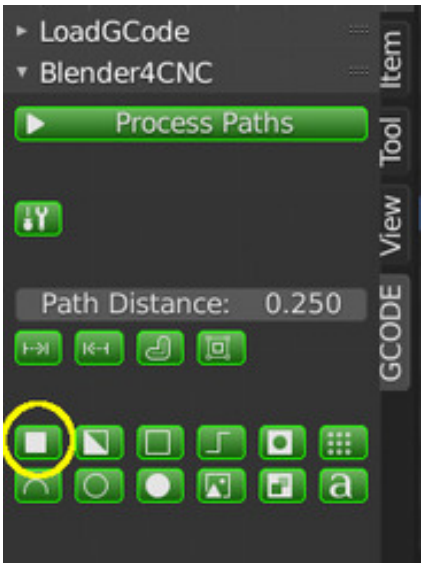

Figure 71: Under the GCode tab, click the "Create CW Pocket" button.

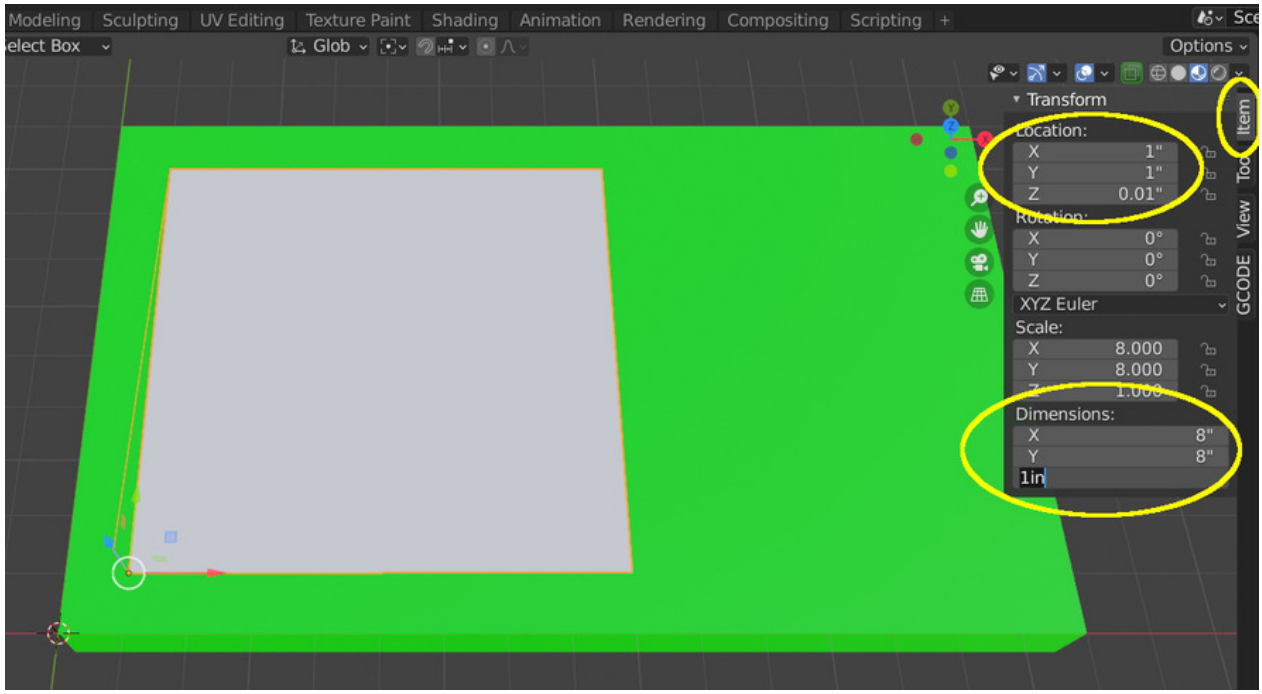

Figure 72: Under the Item tab, select the pocket operation and change the location to (1,1) and this time change the dimensions to X=8 and Y=8.

Figure 73: Under the GCode tab, click to select the pocket, then click the "Create Tenon" button.

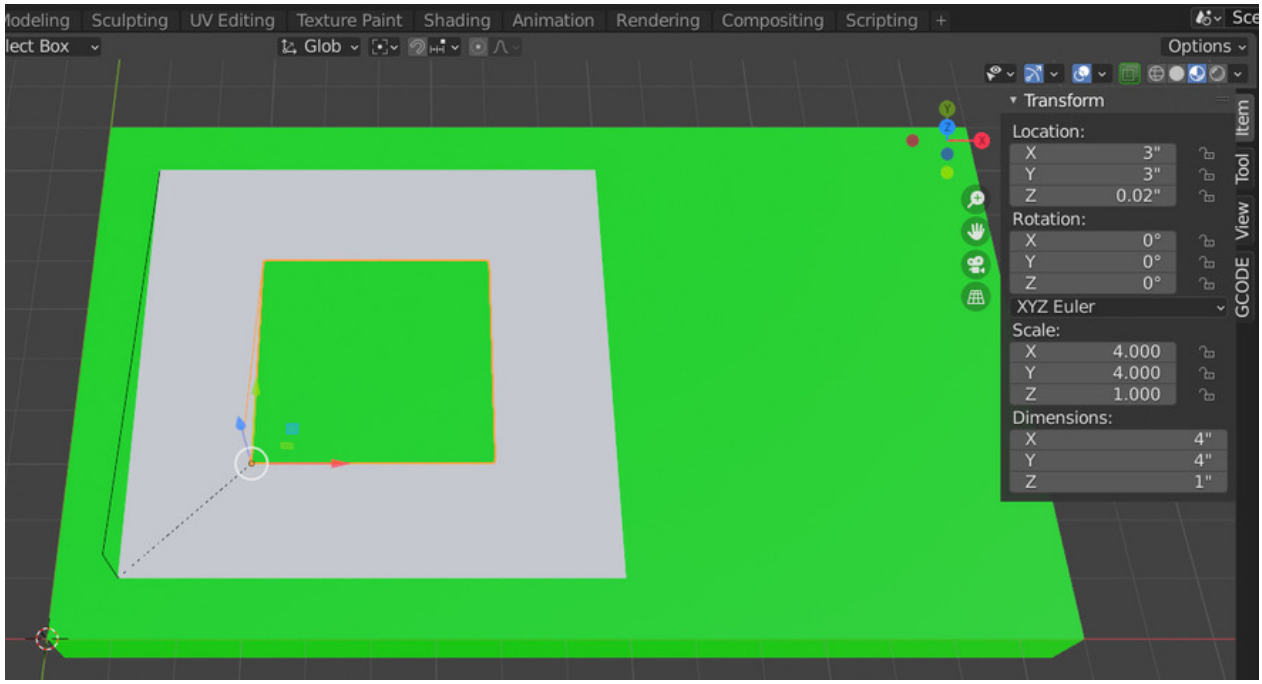

Figure 74: Change the location of the tenon to 3,3 and the dimensions to 4,4.

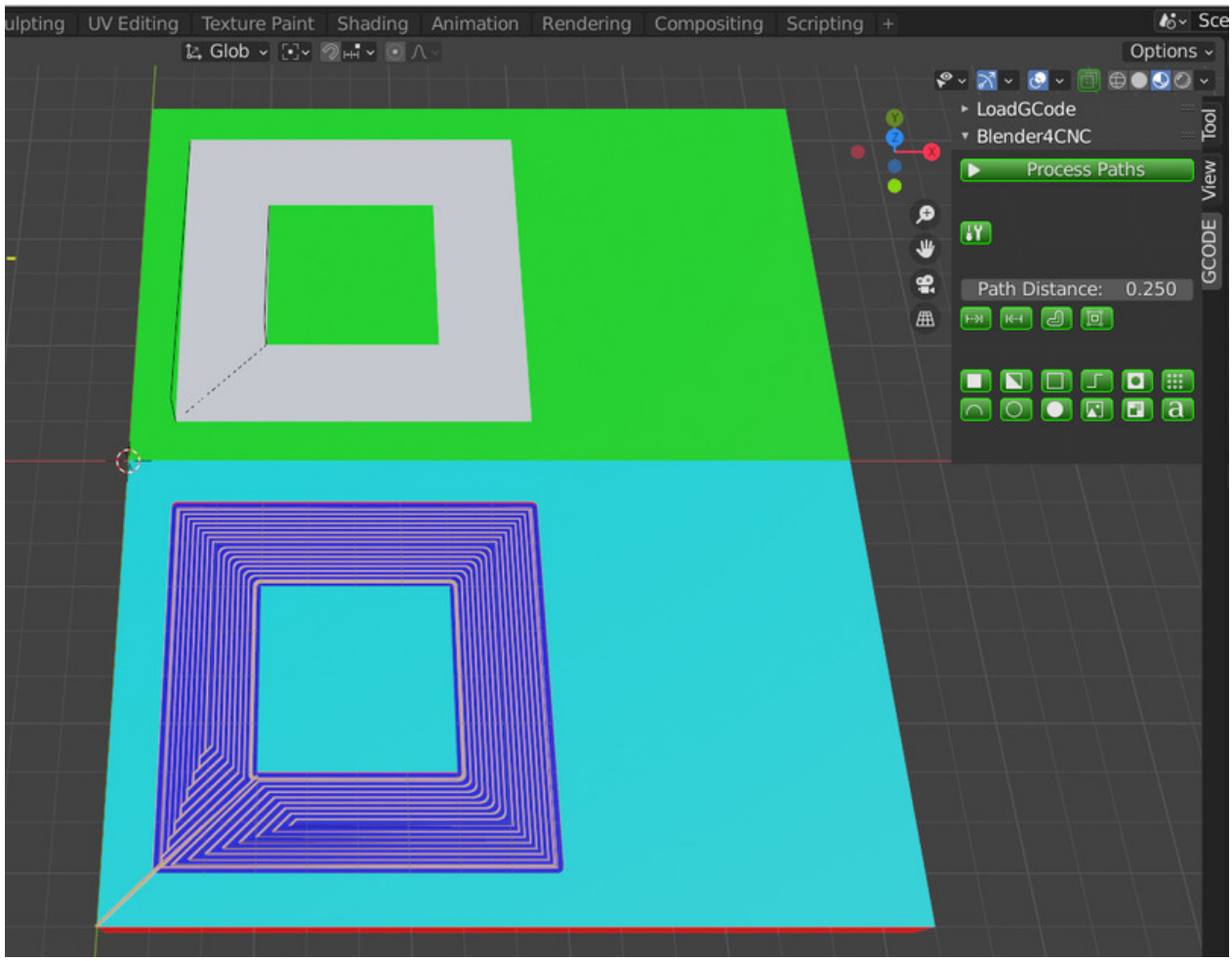

Figure 75: Under the GCode tab, click "Process Paths" to see the result. You may need to use the middle mouse button to zoom out and see all the results.

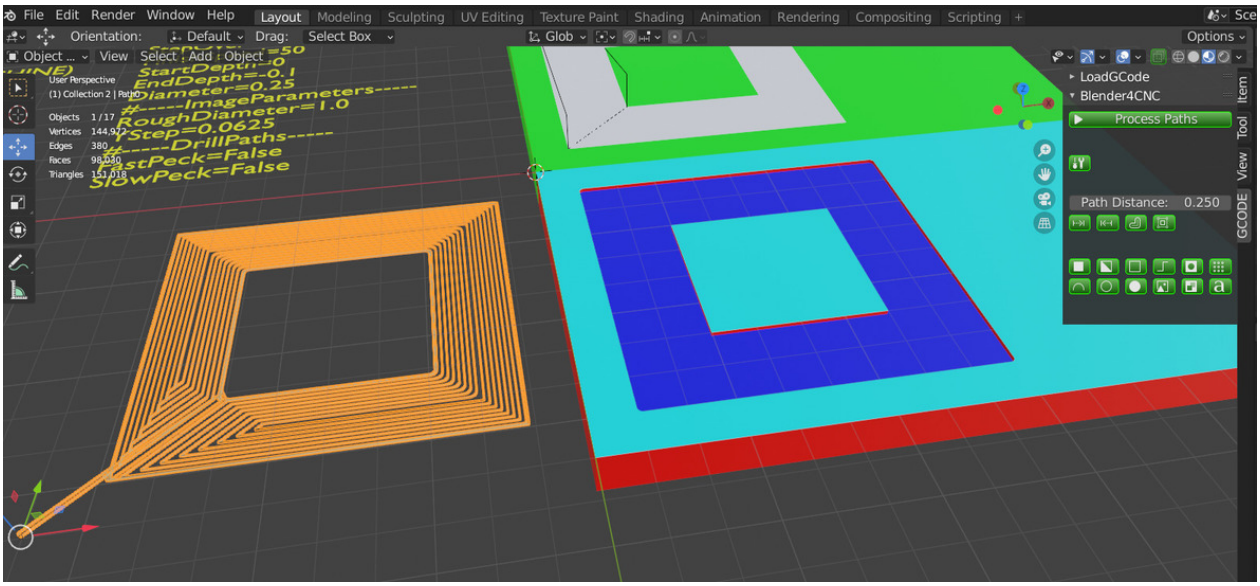

Figure 76: Select the red path object and move to the left for a little more clarity.

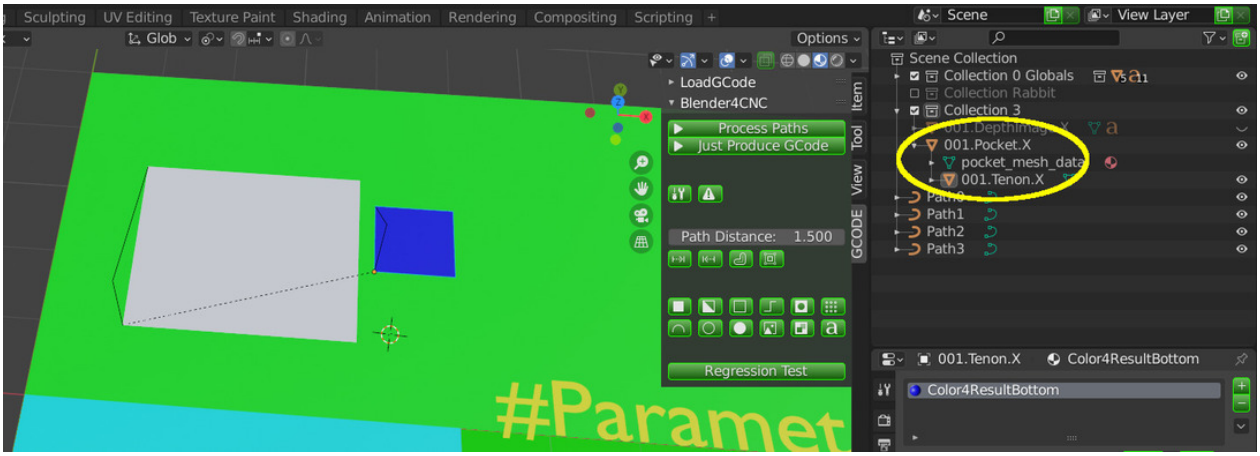

Figure 77: It is an error to place a tenon (blue) completely outside a pocket.

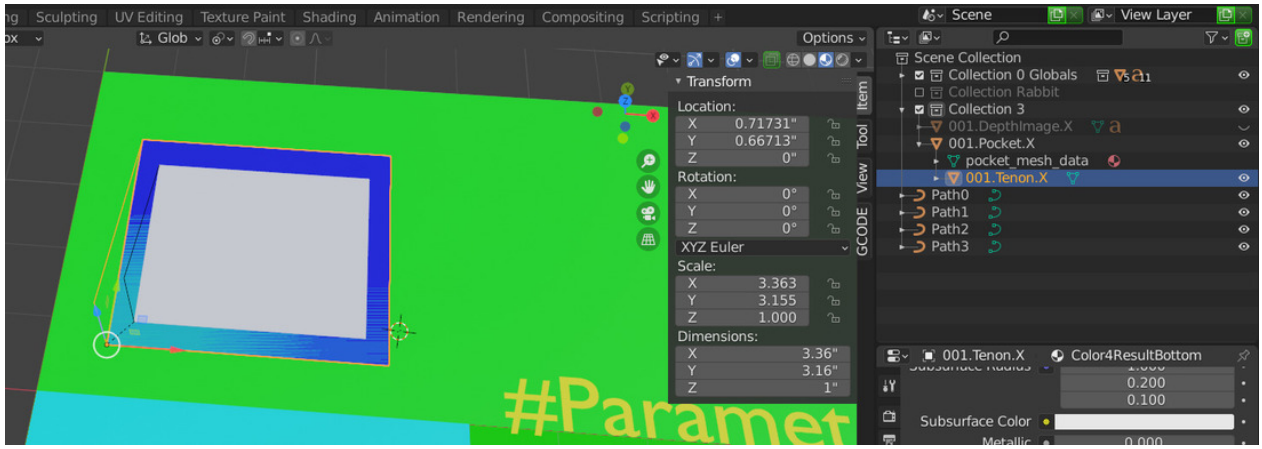

Figure 78: It is an error to make a tenon larger than a pocket.

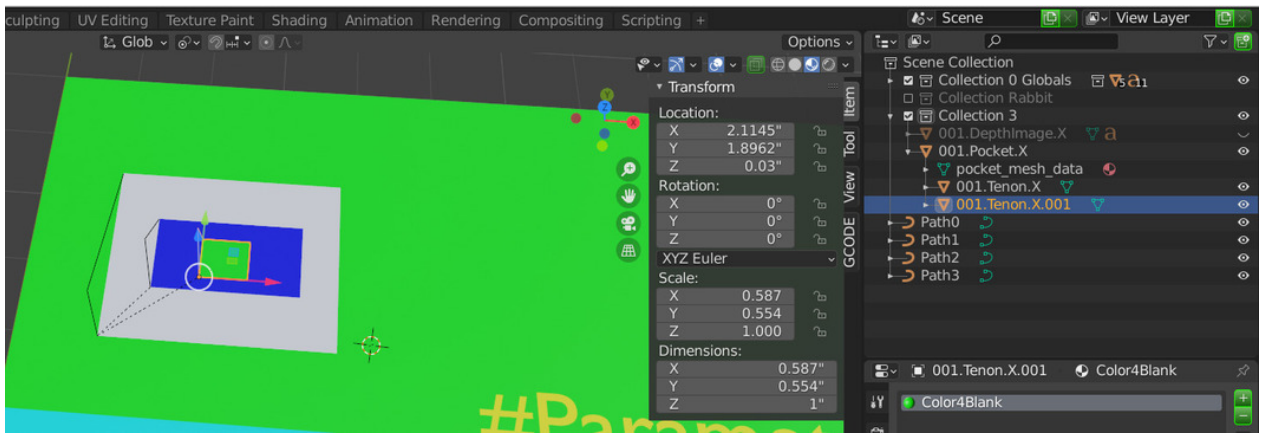

Figure 79: If a small tenon (green) is completely contained inside a larger tenon (blue) it will be ignored.

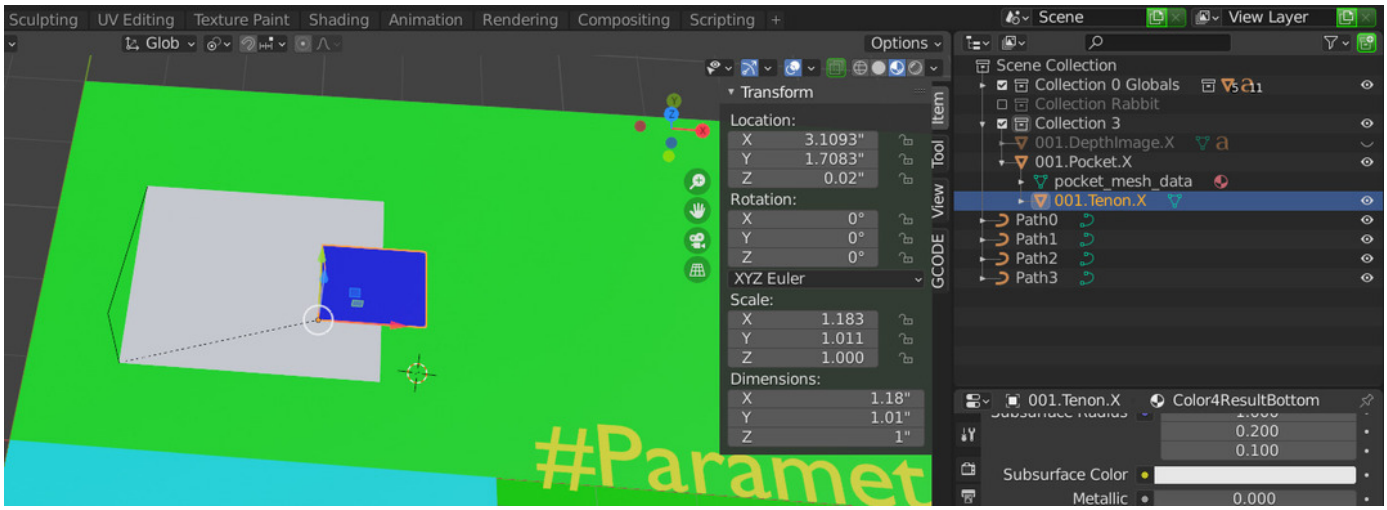

Figure 80: A tenon cannot overlap the edge of a pocket. This will produce an error.

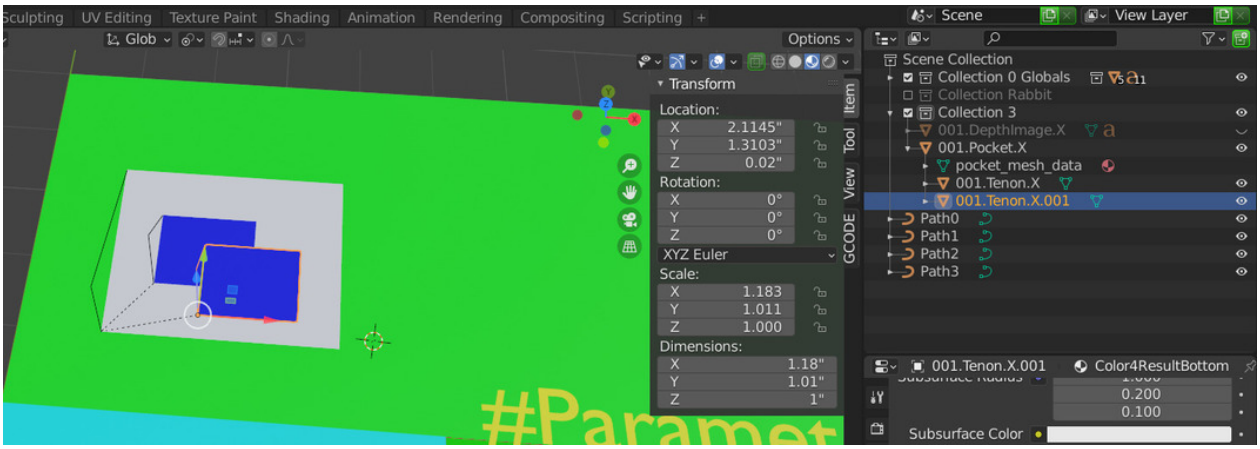

Figure 81: Tenons cannot overlap each other. This will produce an error.

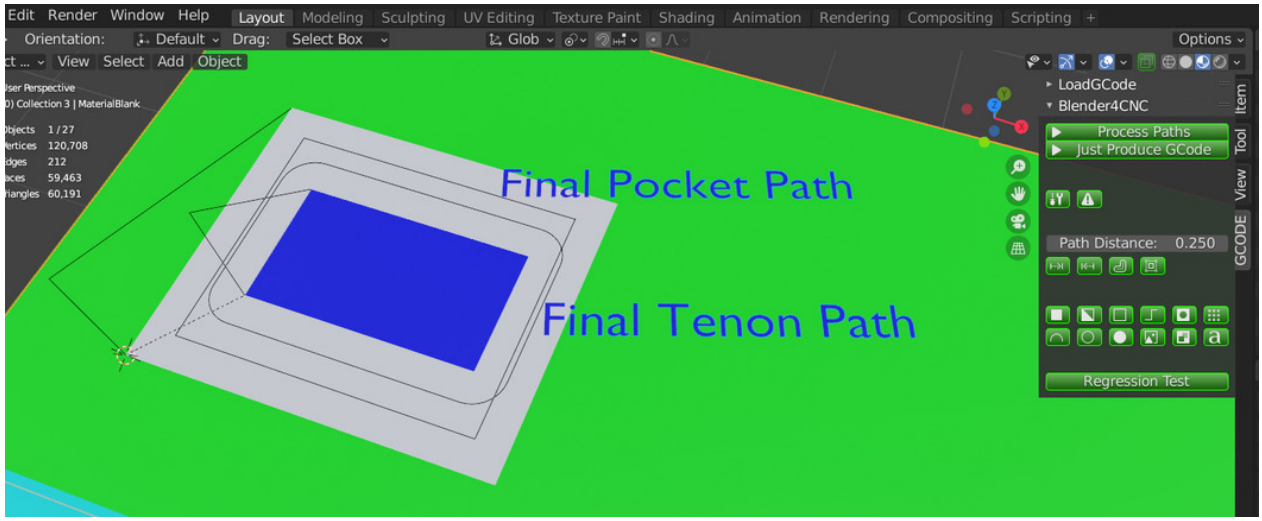

Figure 82: Even if tenons and pocket edges do not overlap, if the paths taken by the cutter to cut the final shape overlap each other then this too is also an error. Make sure there is enough room between tenons and between tenons and pocket edges that the cutter can safely pass between them.

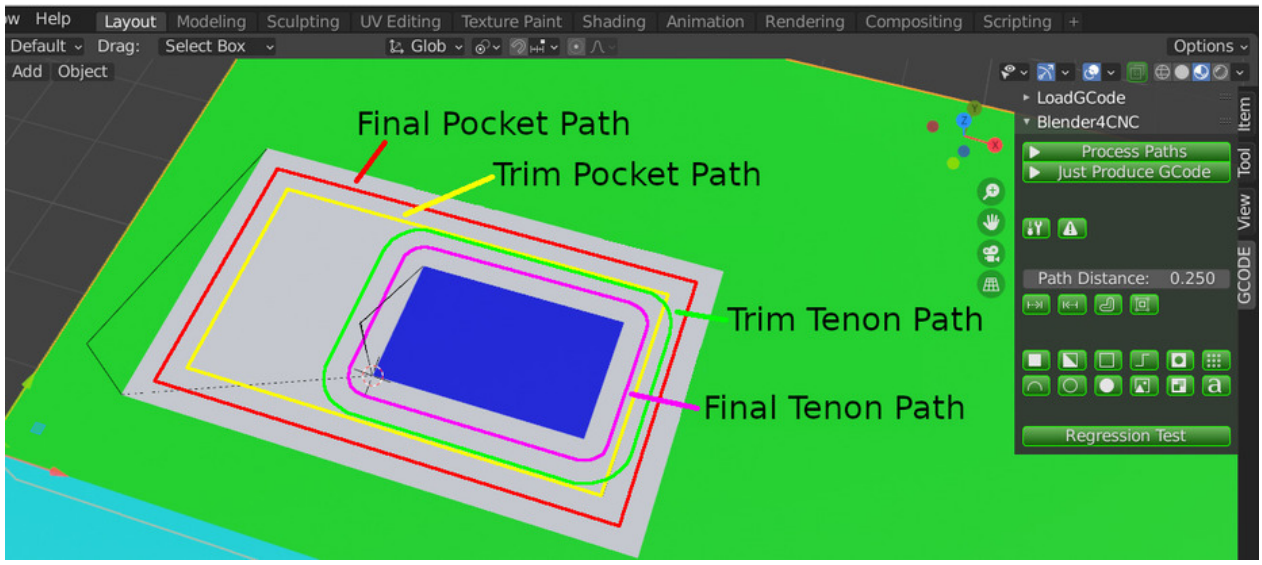

Figure 83: When adding finishing or trim passes, the paths taken by the cutter to cut the finish/trim shape must not overlap each other. This too is an error. Make sure there is enough room between tenons and between tenons and pocket edges that the cutter can safely pass between them even when performing the finish or trim passes.

1.3.4 Quickly Create Operations (using a Template) - A Depth Image.

Figure 84: A Depth Image is a 3-dimensional carving; sometimes called a relief carving. (Left - carved from exotic woods; middle - carved from pine.) It is produced from a suitable grayscale image (right). Sometimes these grayscale images are called a height map. This example is of Uluru, a rock formation in Australia.

1. The Multiple passes of a Depth Image

Figure 85: To minimize the time taken to smoothly carve a depth image it is performed in 4 stages using different router bits.

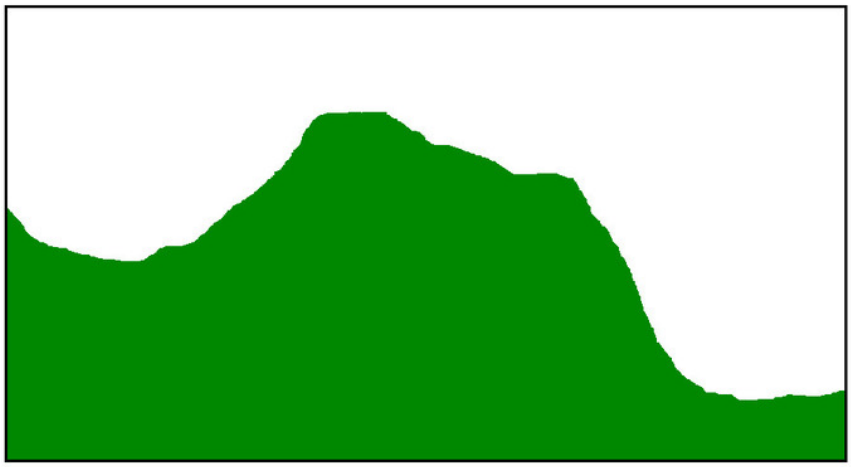

Figure 86: Cutting a depth image uses 3 router bits and cuts in 4 phases. The first bit (used in the first phase) will be a large diameter, flat nose bit. The final bit (used in the 3rd and 4th phases) is a small diameter ball nose bit. The second bit is a flat nose bit of the same diameter as the final bit. Imagine that the above image represents the 2-dimensional profile of a depth image. We want to cut this profile as closely as we can and as quickly as possible. Blender4CNC produces GCode to do this in 4 phases. (Which means it produces 4 separate GCode files.)

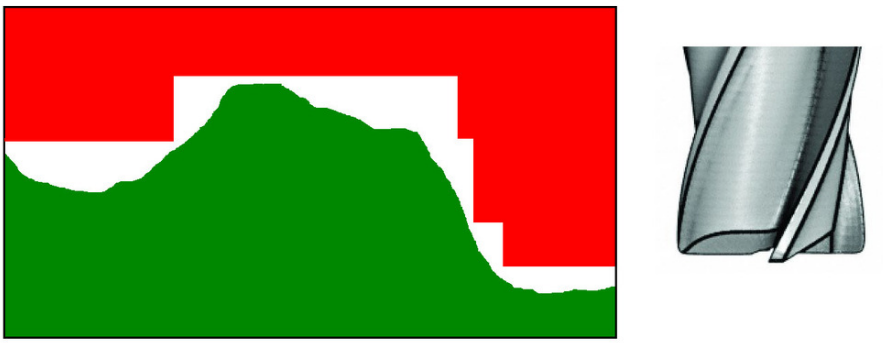

Figure 87: In the first phase, a large, flat router bit (maybe 1" or 25mm) is used and it very quickly removes a lot of material but it can't get very close to the exact profile.

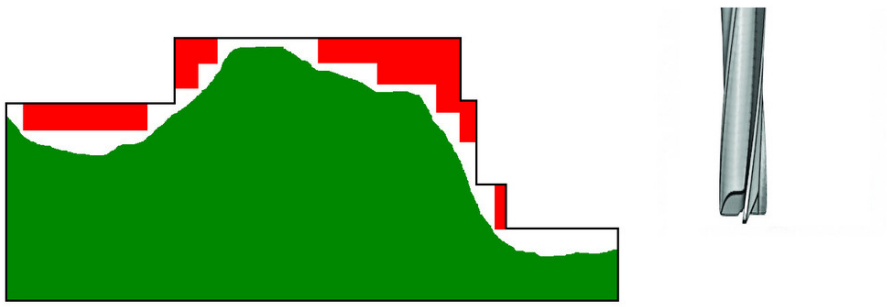

Figure 88: In the second phase, a flat router bit of the same diameter as the final router bit (maybe 0.25" or 6mm) is used and it very quickly removes some material and can get closer to the exact profile.

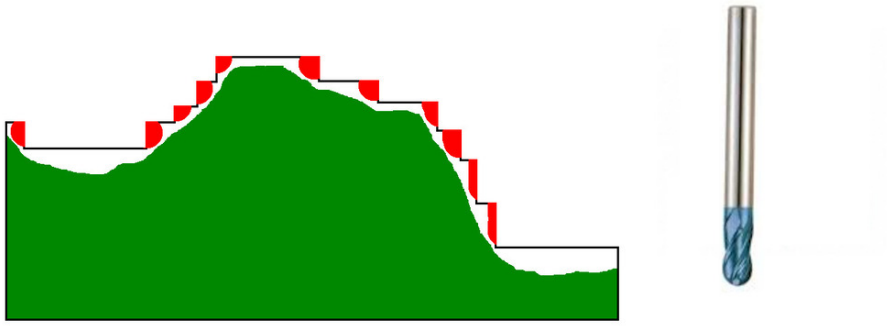

Figure 89: In the third phase, the final router bit (a ball nose router bit of maybe 0.25" or 6mm) is used and it very quickly removes some material and can get even closer to the exact profile.

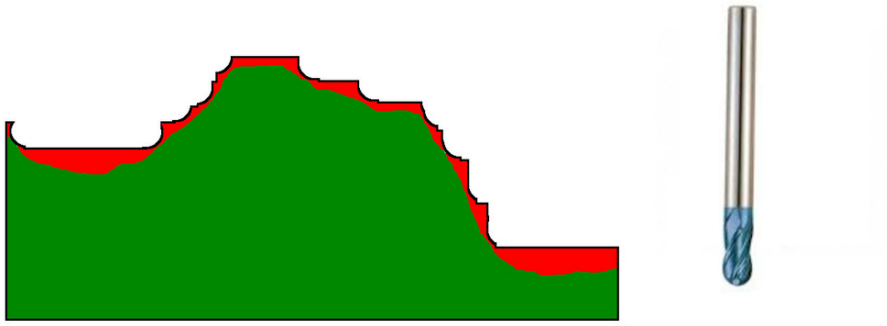

Figure 90: The fourth phase takes a really long time. It must go back and forth along the X axis for the full length of the image and slowly move up the Y axis for each new line until it has "rasterized" the whole image. It may do this at a slower speed and because of all the details in the profile it may never reach the maximum speed as it accelerates and deccelerates to change directions so often. But, because the previous phases removed a lot of material, the 4th phase only needs to be done once.

2. Depth Images - Grayscale Height Map

   To create a Depth Image on a CNC router you need a grayscale image. That is an image that only contains pixels that are shades of gray (black and white are the darkest and lighest shades of gray). The blackest pixels represent those parts of the image where you want the router bit to cut the lowest and the lightest pixels represent those areas of the image where you don't want to cut deep (white pixels mean the top of the surface and so don't even need cutting). The darker a pixel is, the deeper the cut at that location. For example, a 50% gray pixel means that the router bit will plunge to half depth at that location. Let's look at some simple examples to get an idea.

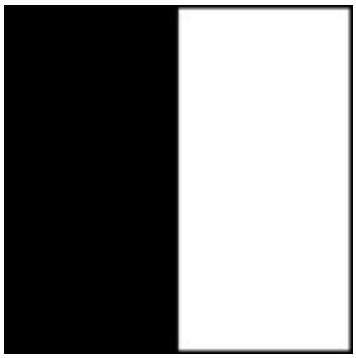

Figure 91: Here is a grayscale image that happens to only contain pure black and white pixels.

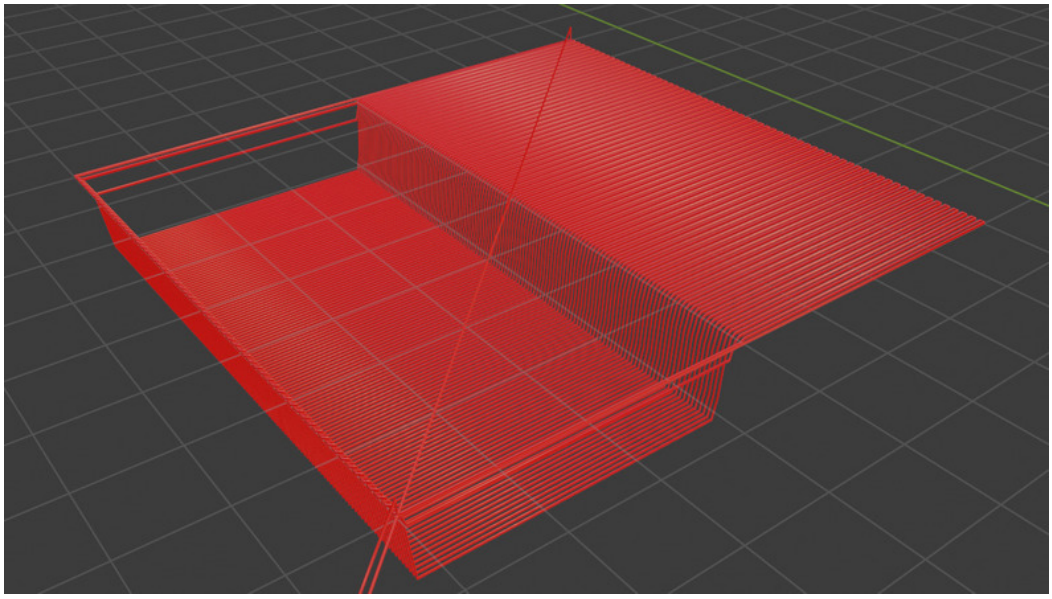

Figure 92: Here is how the depth image will be carved for the image in figure 91. It cuts to the full depth where the image is black and leaves the material at the full height where the image is white.

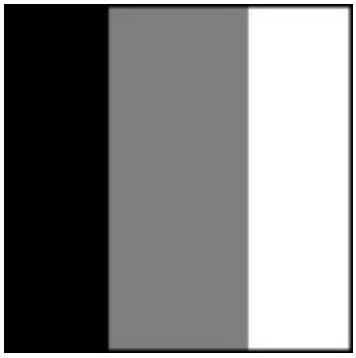

Figure 93: Now we add a 50% gray bar in between the black and white halves.

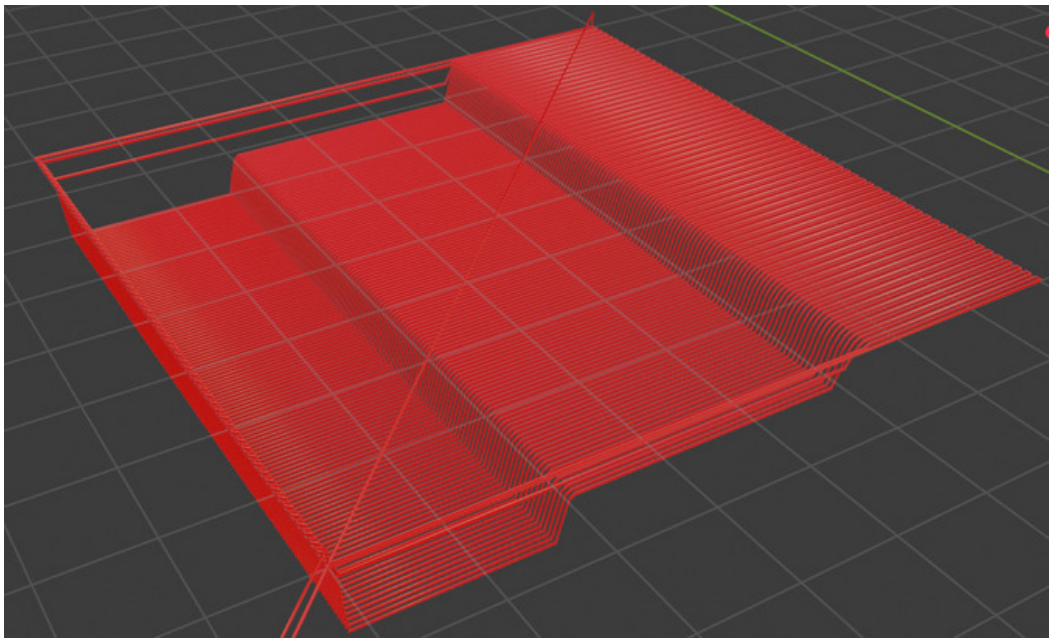

Figure 94: Here is how the depth image will be carved for the image in figure 94. The area of gray pixels is about half as high as the white pixels.

Figure 95: Here is a grayscale spiral.

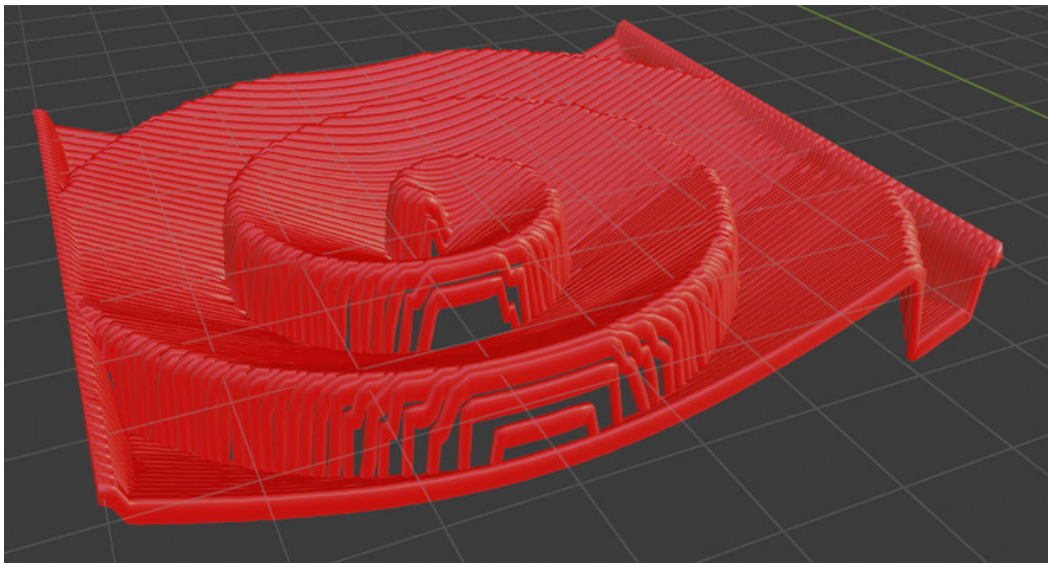

Figure 96: Here is how the depth image will be carved for the spiral.

3. The "Resolution" of a Depth Image

Figure 97: A depth image is strongly affected by the "YStep" parameter. This defines the "resolution" of the carving in the Y axis **and** the X axis. A smaller YStep will result in more lines in the carving and this gives a smoother result (better resolution) and takes a longer time. This is changed in the "Parameters" text field on screen.

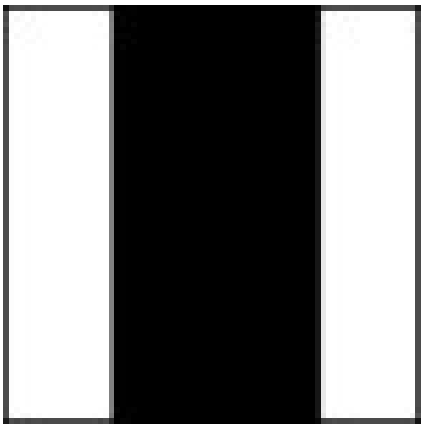

Figure 98: Here is a little test image that will demonstrate how the YStep changes the resolution in both X and Y axes.

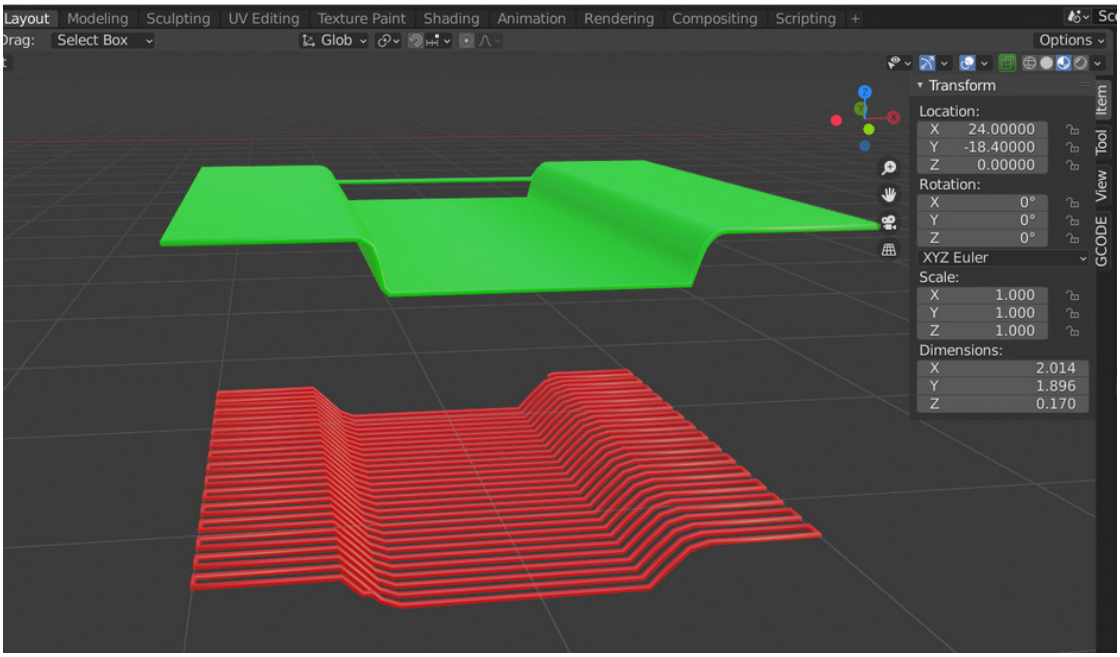

Figure 99: This image shows 2 paths for a cutter to carve the relief shown in figure 98. The top path in green was produced for a YStep of 0.015625 (inches) and looks quite smooth. The lower path in red was produced for a YStep of 0.0625 (inches) and is much coarser showing the rows quite clearly.

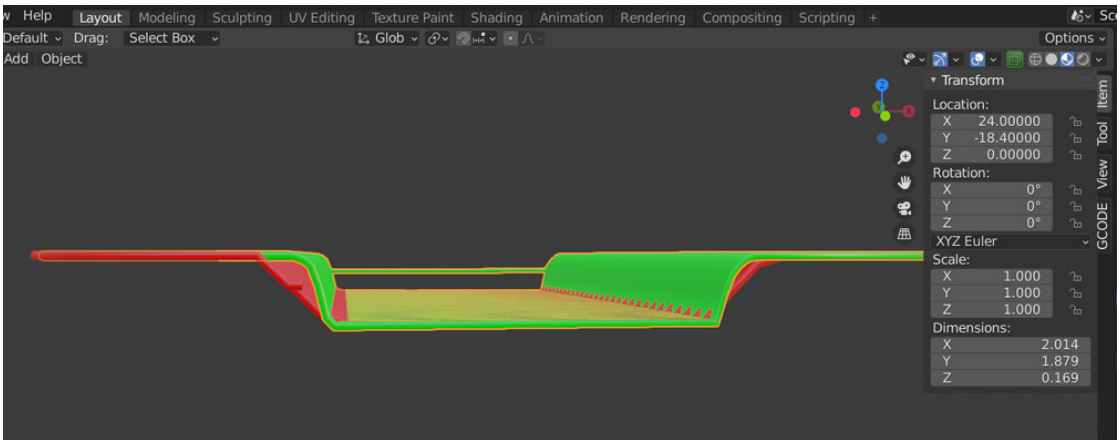

Figure 100: Look at the two paths when we overlay them on top of each other. You can see how the coarse red path also has a much coarser resolution along the X axis. The green path with the fine resolution can follow the edge (from white to black) much better. The red path actually removes more material at the edge - it slants down much earlier than the green path in a straight, diagonal line to the level of the black.

4. Depth Image Parameters

Table 2: Table of Depth Image Parameters. These are the main parameters that affect how a depth image is carved. The example values are for a machine that is imperial (inches).

Name	Description	Example
SafeZ	A safe Z height where the tool will not hit the material when performing rapid movements (G0 codes)	0.1
ZStep	When cutting rough passes, the generated GCode will step in the Z axis by this amount for each depth pass.	-0.1
Speed	Rapid and cutting movement speed.	60
EndDepth	The final depth of a pocket.	-0.5
Diameter	The diameter of the router bit.	0.25
RoughDiameter	The diameter of the router bit to be used for the first rough pass. Usually large to remove a lot of material quickly. Only the first stage of cutting a depth image will use this router bit.	1.0
YStep	The distance between "lines" in a depth image. A depth image is produced by cutting many lines very close to each other to produce a smooth result.	0.0625

5. Creating a Depth Image CNC Operation in Blender4CNC

   Open the template project named "Template-Imperial.blend". Go to "File", "Save As" and save the project with a new name (so you don't accidently overwrite the template file). Click on "Collection 2" in the upper right window of the display to make sure it is selected.

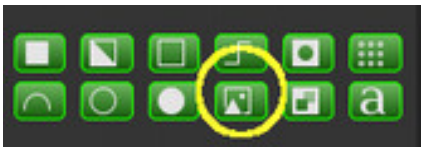

Figure 101: Click the "Create DepthImage" button.

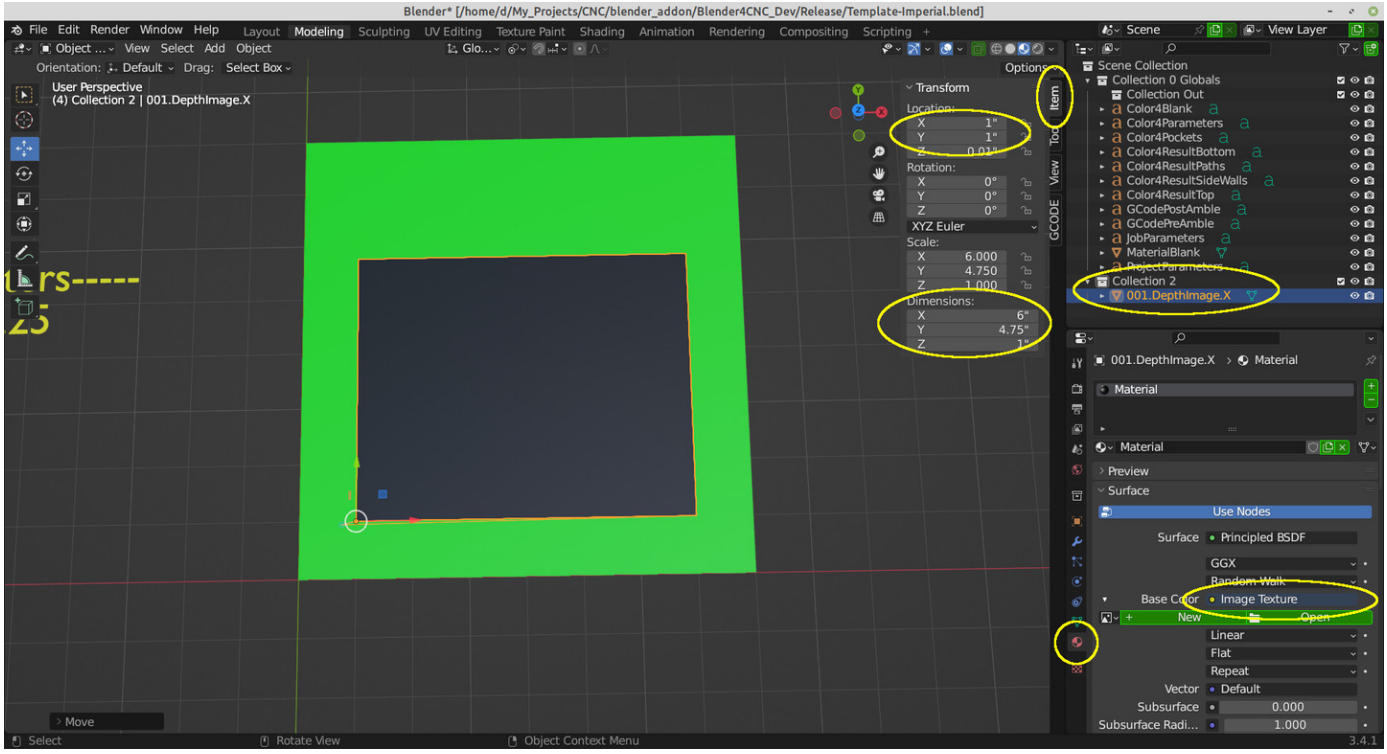

Figure 102: Under the Item tab, change the depth image location to 1,1 and set the dimensions to X = 6, and Y = 4.75. If you click the "material" tab in the lower right sub-window you will notice that a material has been added to this object and the "Base Color" is set to "Image Texture".

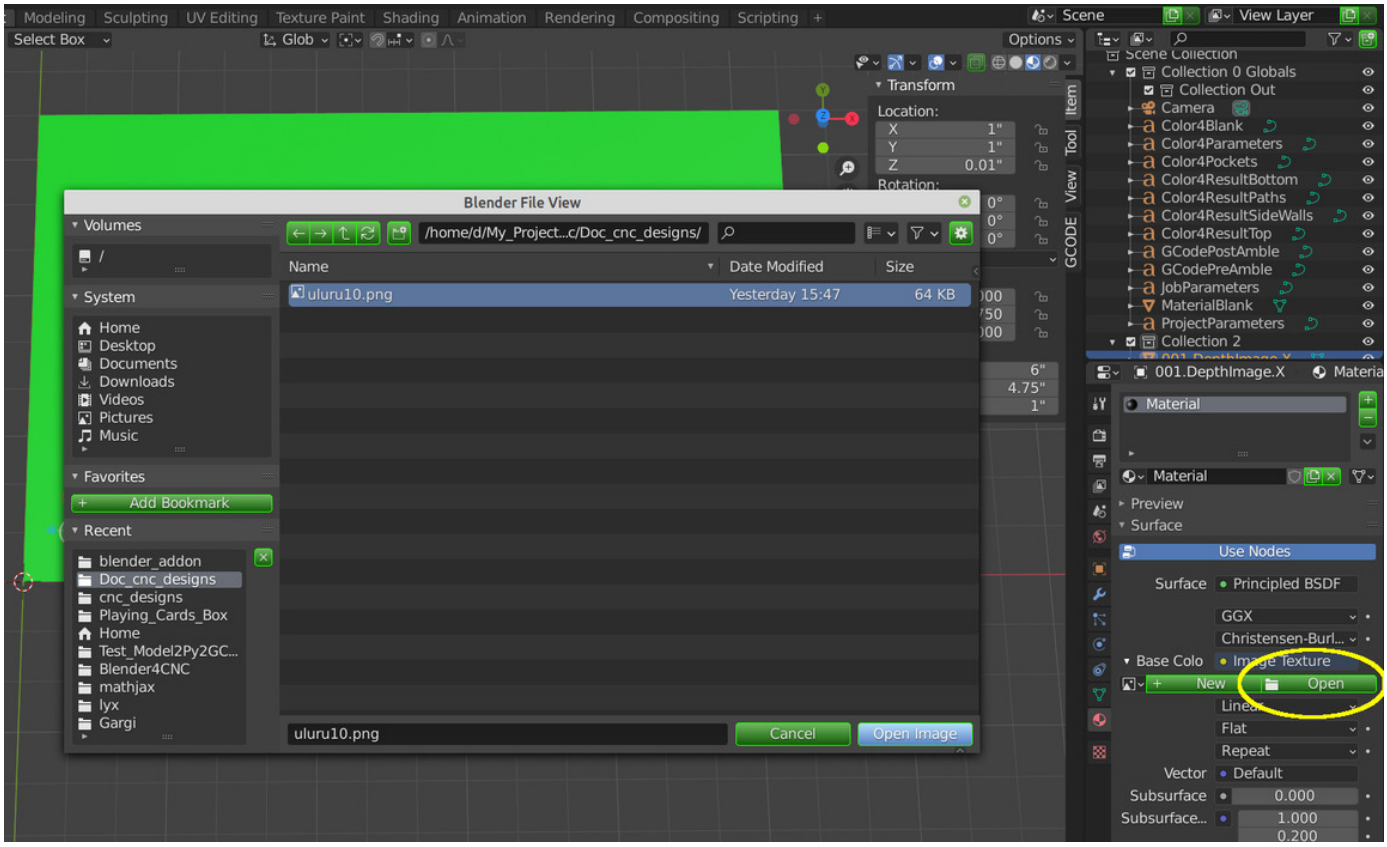

Figure 103: Click the "Open" button under the material in the lower right corner of the display and then browse to and select an image(an example image "uluru10.jpg" comes with Blender4CNC).

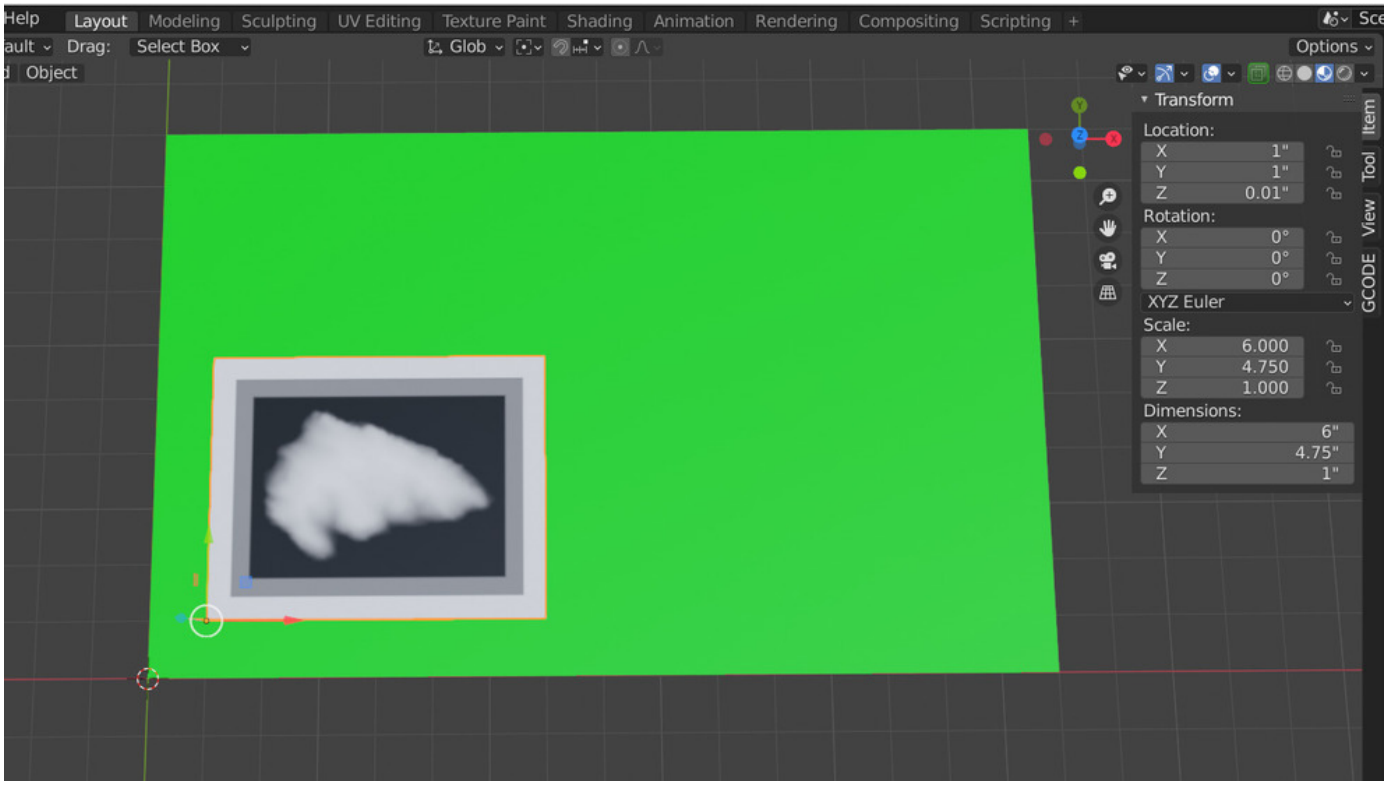

Figure 104: You should see the image appear in the view.

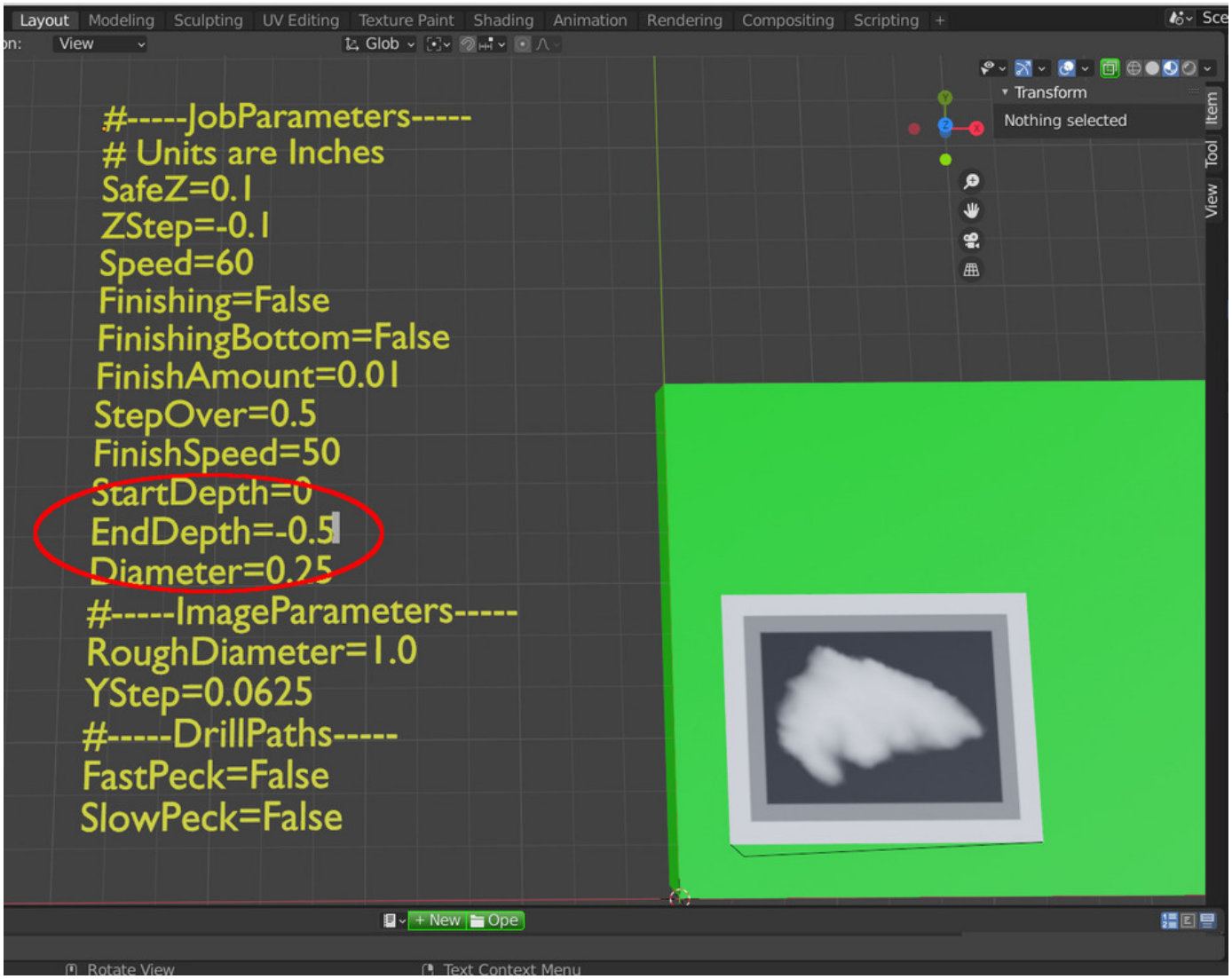

Figure 105: Before we see the result, change the depth in the Job Parameters. Select the "JobParameters" text object and press TAB to enter "Edit Mode". Change the value for "EndDepth" to -0.5 and then press TAB again to exit Edit Mode.

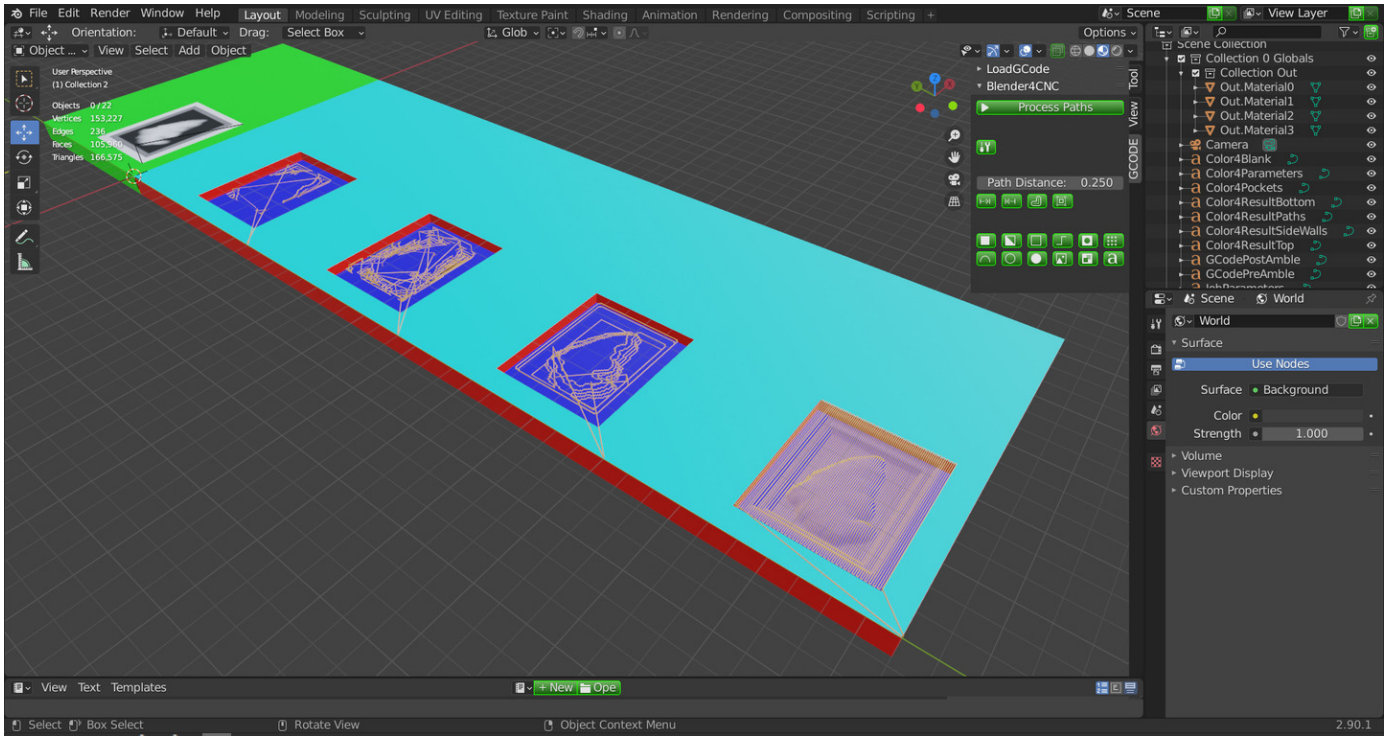

Figure 106: From the GCode tab, click "Process Paths" to see the result. A depth image is cut in 4 passes. These passes are predefined and cannot be changed. The first (rough) pass is done with a large diameter bit (in this example 1.0"). The second (rough) pass is done with a bit that has the same diameter as the final bit (in this example 0.25"). The third (rough) pass and the final pass are done with the final spherical bit. See the section "The Multiple passes of a Depth Image" for more explanation.

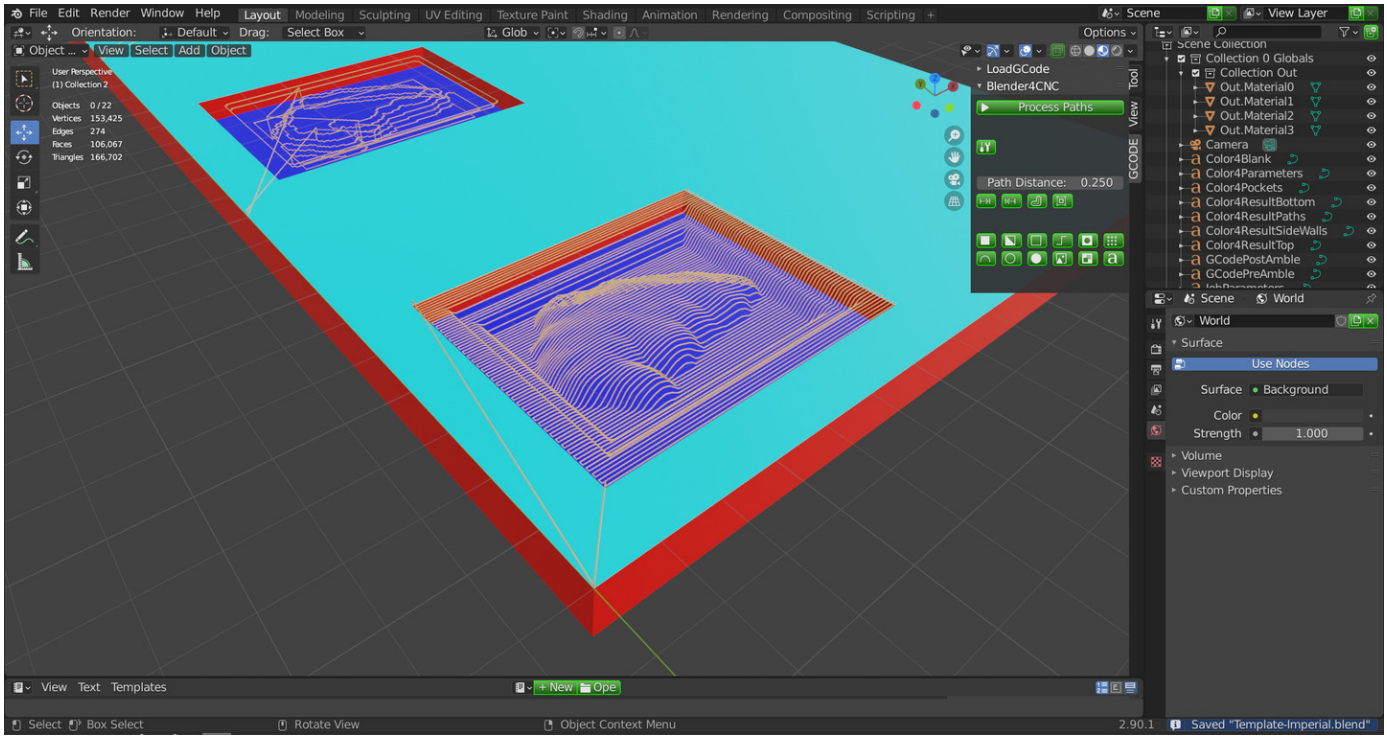

Figure 107: Here we have zoomed in on the final pass to show the detail of the depth cuts.

1.4 An Example of a Depth Image (Relief Carving) with Decorations

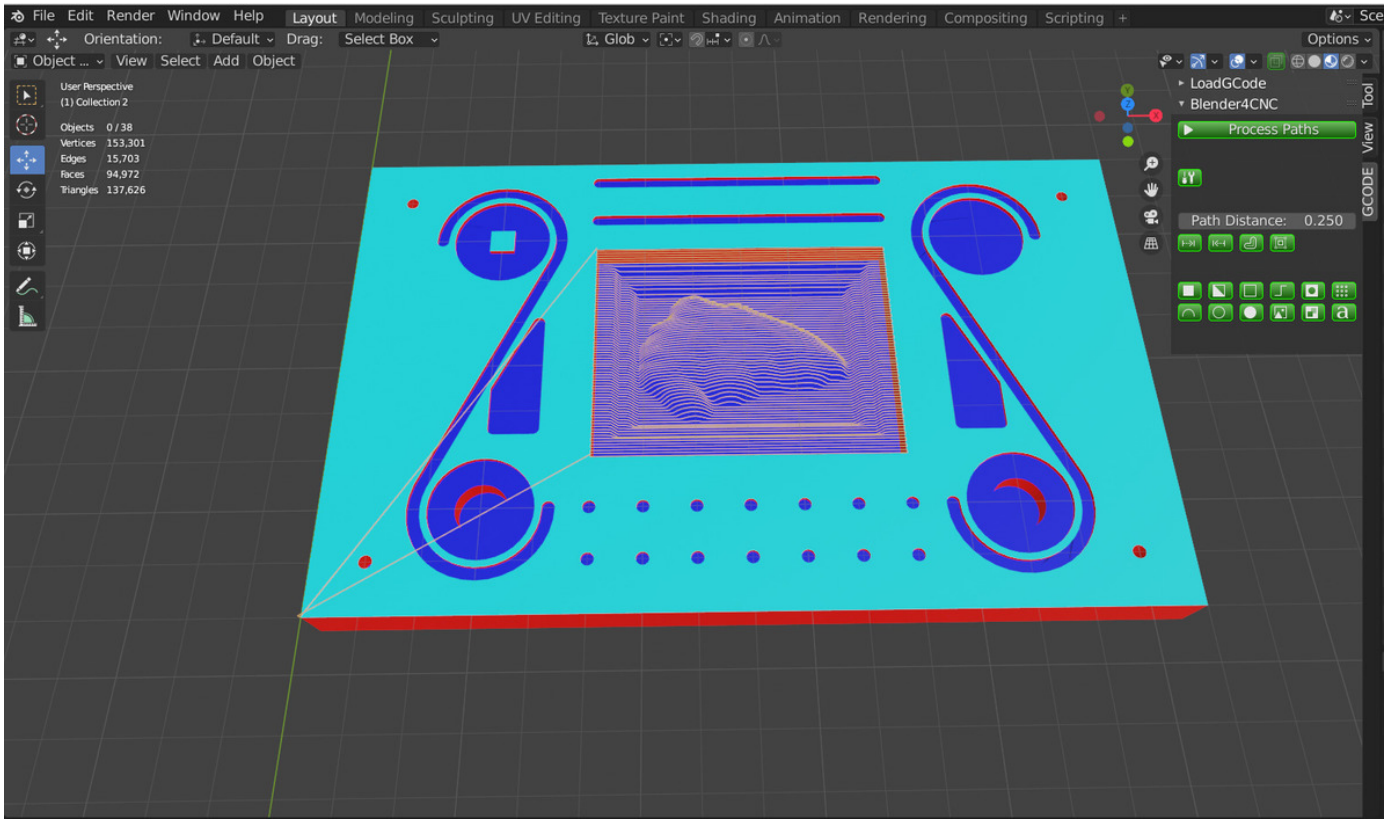

Figure 108: Here is an example demonstrating many of the features in Blender4CNC - paths, pockets, tenons, circular pockets, drilling holes, depth image carving, multiple phases, curves, straight lines, job parameters, layer parameters, tenons and operation specific parameters.

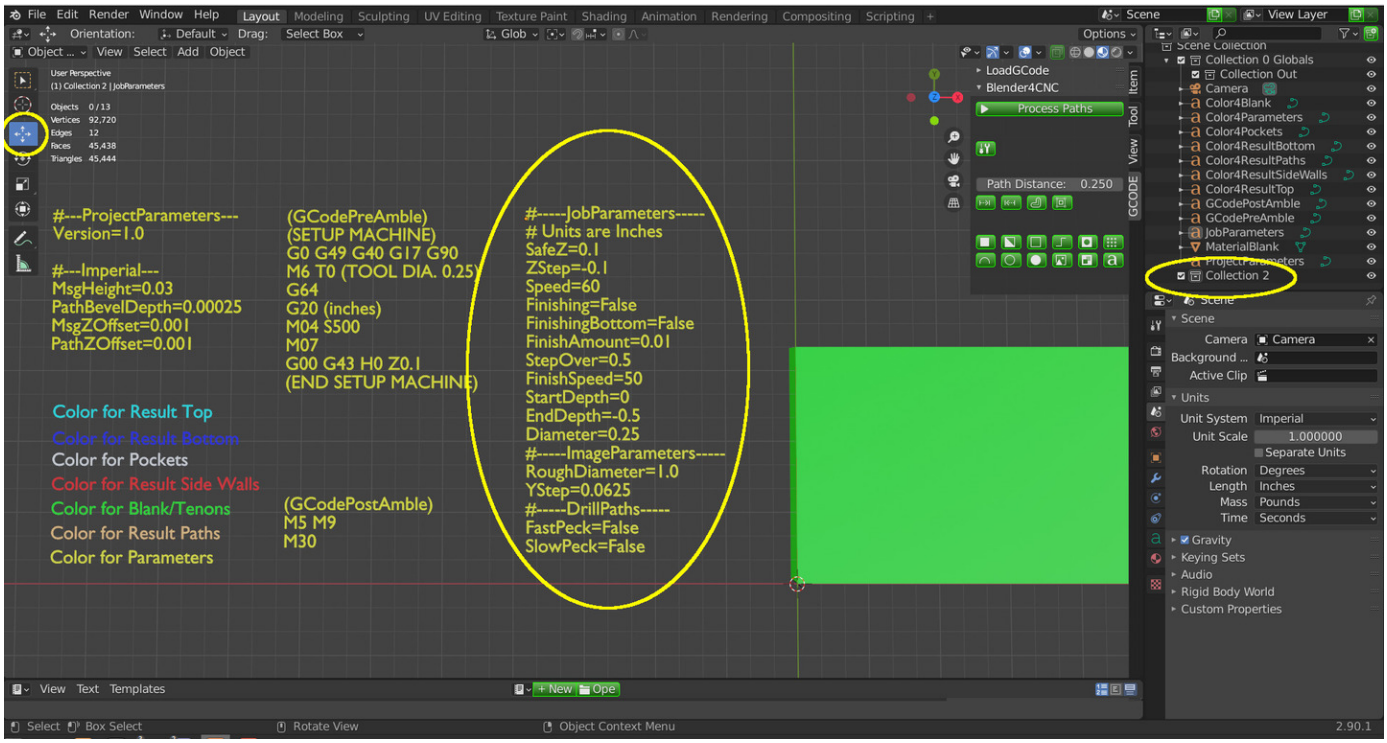

Figure 109: In Blender, open the template project named "Template-Imperial.blend". Go to "File", "Save As" and save the project with a new name (so you don't accidently overwrite the template file). There are text objects defining the GCode PreAmble and PostAmble (change these as necessary for your CNC machine). Set the Job Parameters as shown (left-click to select the text object, then press TAB to enter edit mode). There is a material blank, if it is not set to a size of 16"x10"x1", then select it and change the size (in the item tab). In the top right window, there is an empty collection named "Collection 2". Click on "Collection 2" to make sure it is selected. Make sure to also select the "4-way arrows: icon on the left side of the display (this is to make sure we get a red/green/blue "origin" symbol on each object when we select an object).

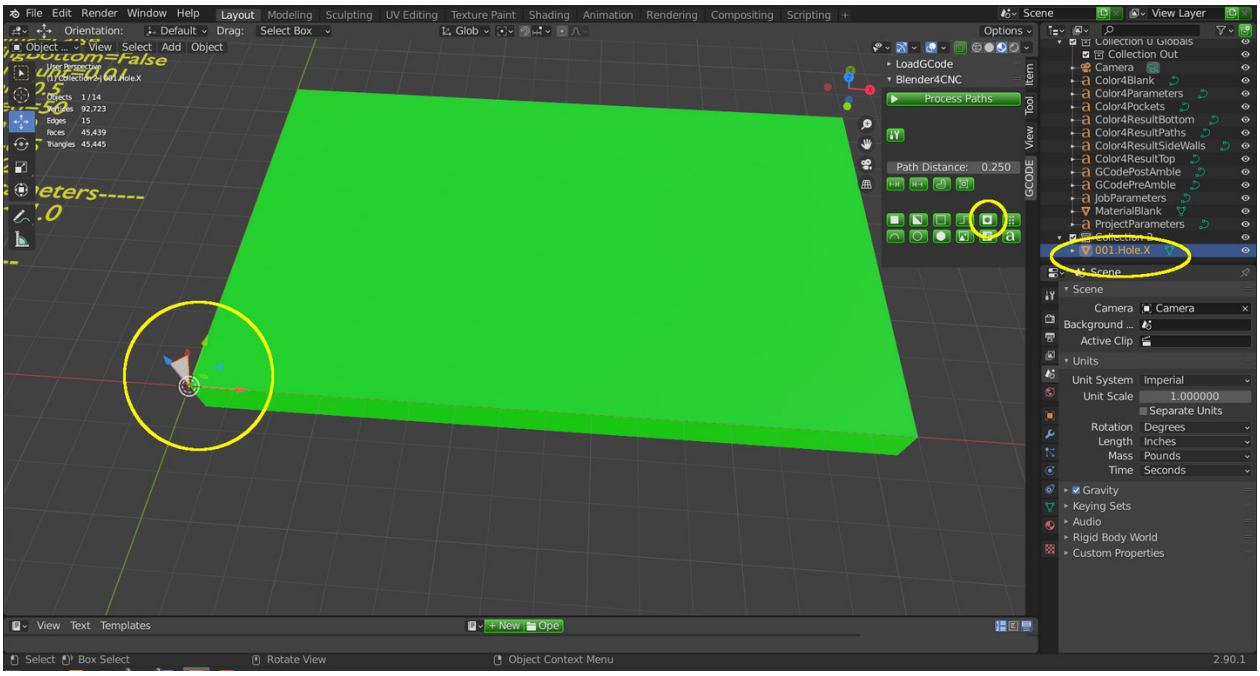

Figure 110: We shall begin with creating 4 holes in each corner. Under the GCode tab, click the "Create Hole" button and a drill hole operation will appear called "001.Hole.X".

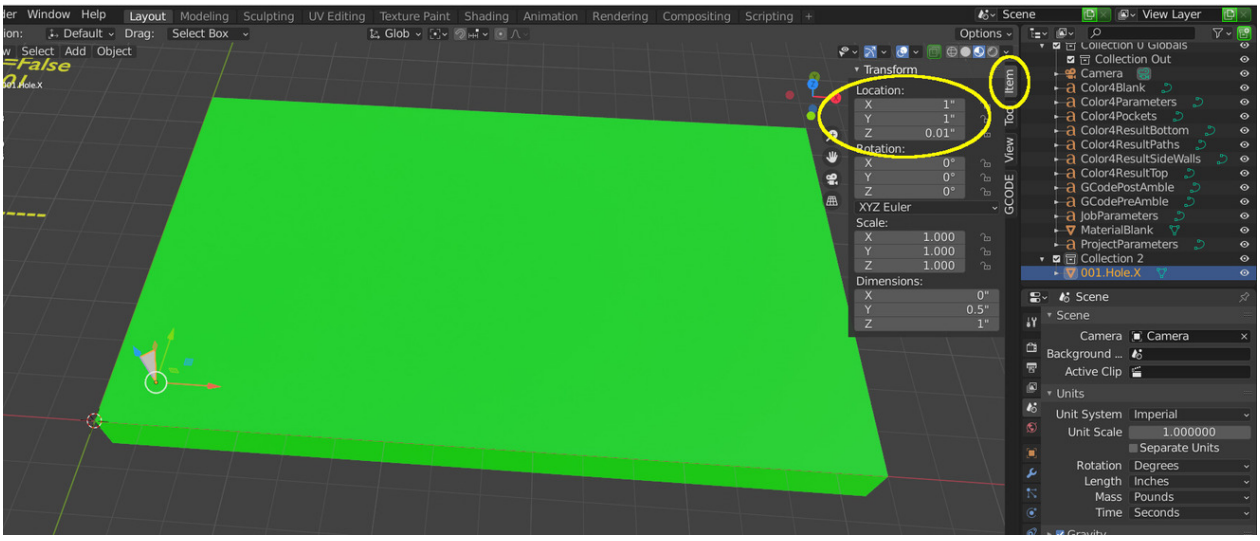

Figure 111: Select the hole operation by clicking on its "flag". Click the head of the red arrow and drag it to X=1.0. If you hold the ctrl key down while dragging, the object will snap to the blender grid and this makes it easier to drag it to exactly 1.0. Repeat with the green arrow to drag the operation up to Y=1.0. Alternatively, you can enter the coordinates directly under the Location in the "Item" tab.

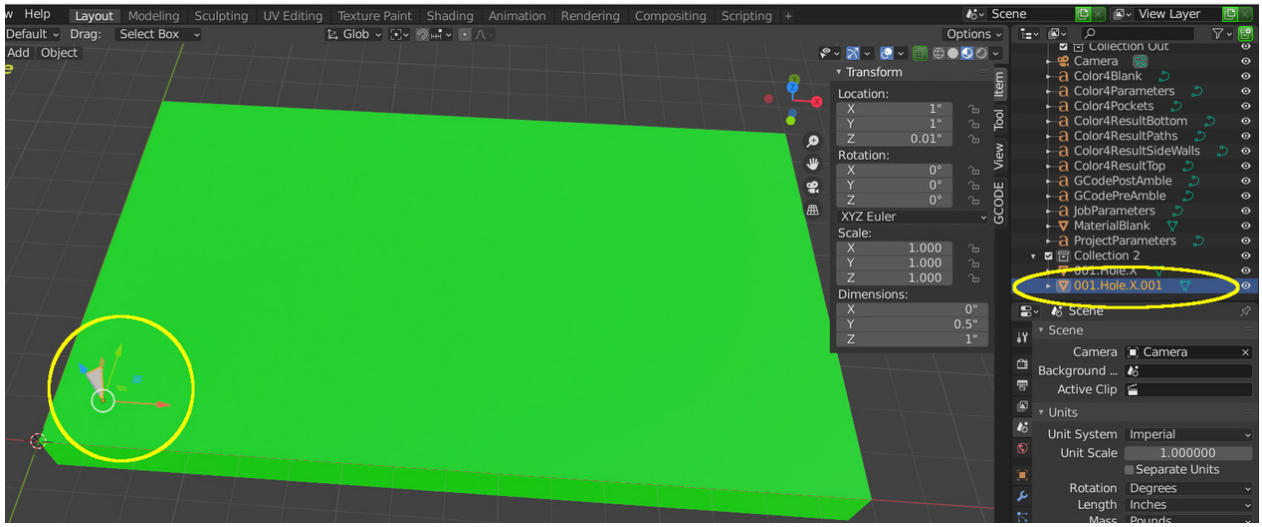

Figure 112: To make the next hole, let's copy the first hole with Blender's "duplicate" function. Select the hole and press "shift-d" then press Enter. You will see another object called "001.Hole.X.001" in the list although the viewport still looks the same as if there is only one operation - that is because the 2nd hole is exactly covering the first hole at X,Y = (1.0, 1.0).

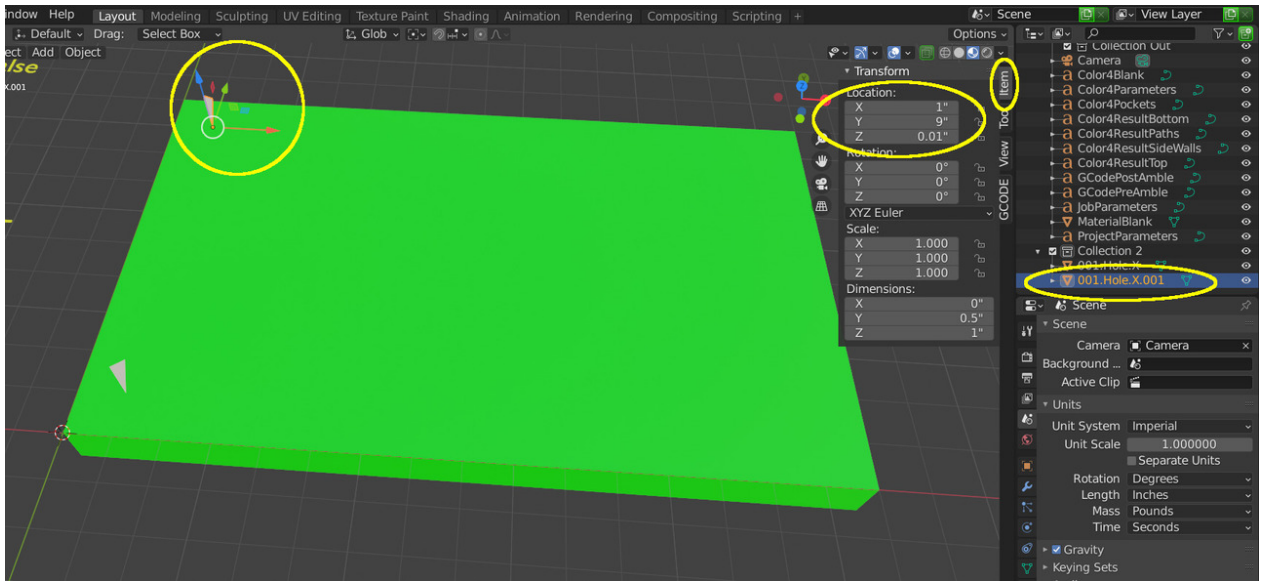

Figure 113: The 2nd hole operation should be the only object selected. Note that you can select an object by clicking on it in the viewport (like clicking on its flag) or you can left-click on it's name in the list of objects at the top right of the display. Move it up to Y=9.0 by dragging the green arrow (holding the ctrl key will allow it to snap to the Blender grid exactly). Or, you can just enter 9.0 directly into the Y field for the location.

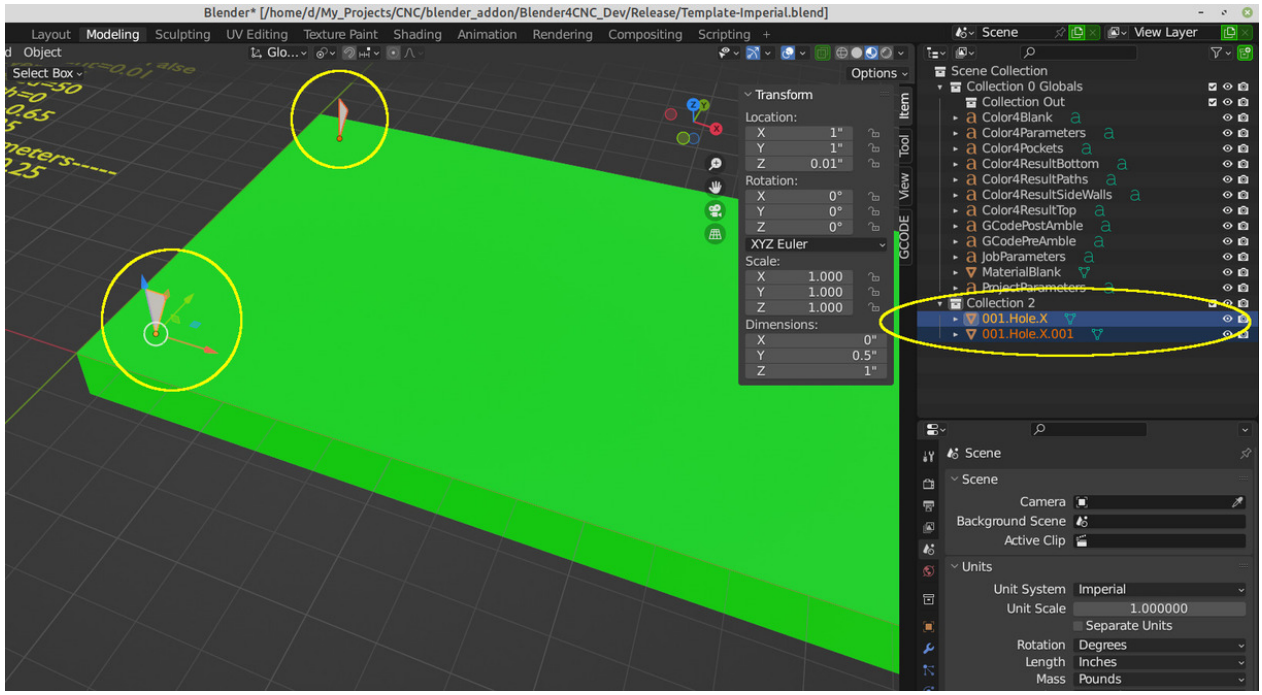

Figure 114: Select both holes. You can either click one hole and then shift-click the other hole in the viewport or you can click and shift-click the names in the list on the upper right.

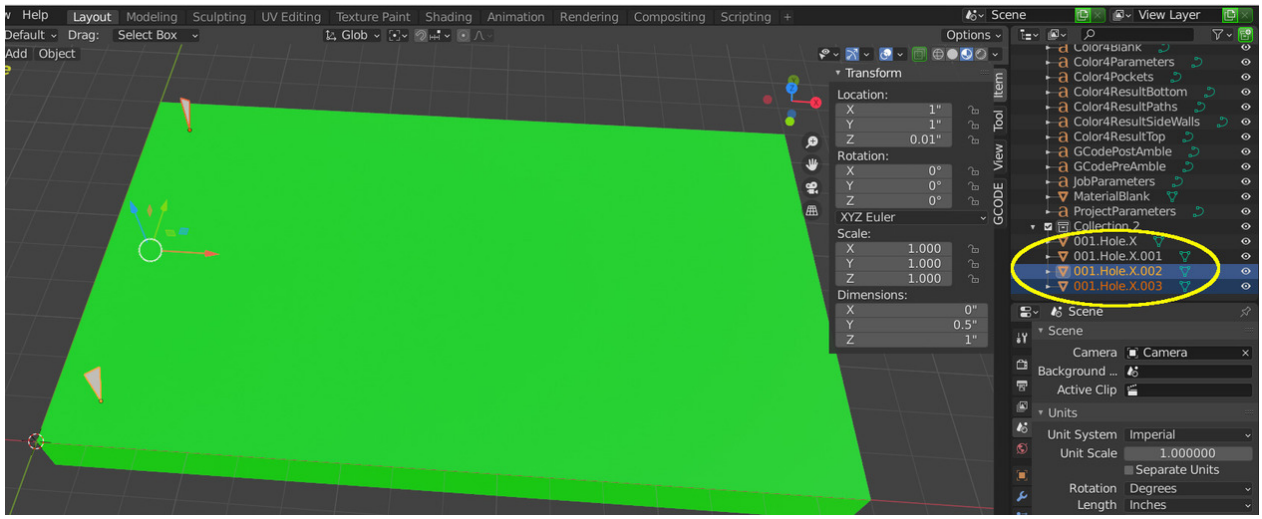

Figure 115: Copy (duplicate) both holes by pressing shift-d then Enter. It will look like nothing has changed in the viewport but there will be two new operations in the list in the upper right. The two new operations are selected (and are covering up the original holes in the viewport).

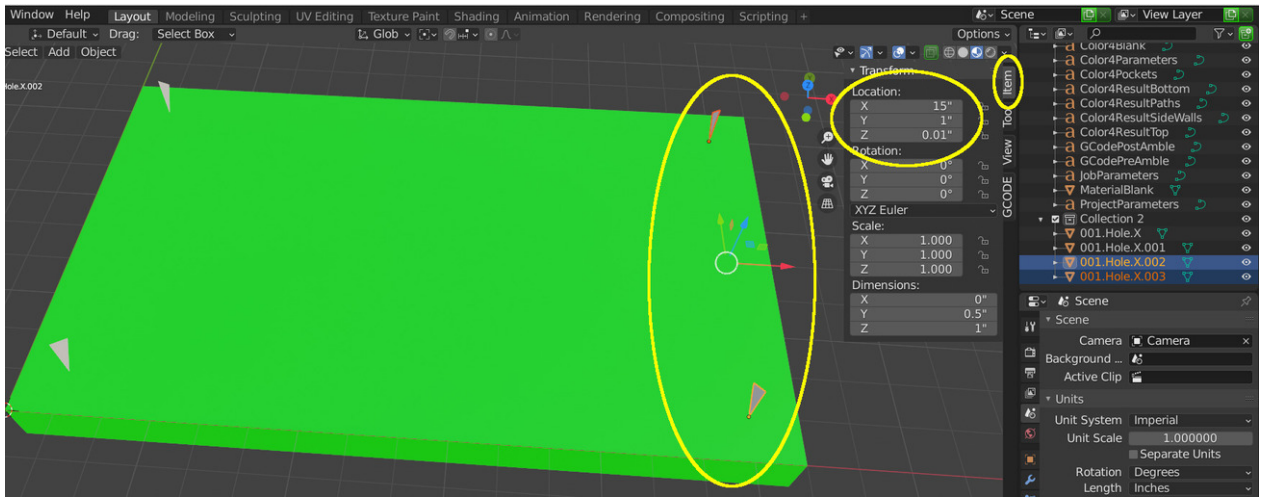

Figure 116: Move the two new holes over by dragging the red arrow (holding ctrl to snap to the Blender grid helps to position exactly on X=15.0 when you get close to the destination). It is also ok to select each hole one at a time and move each hole separately.

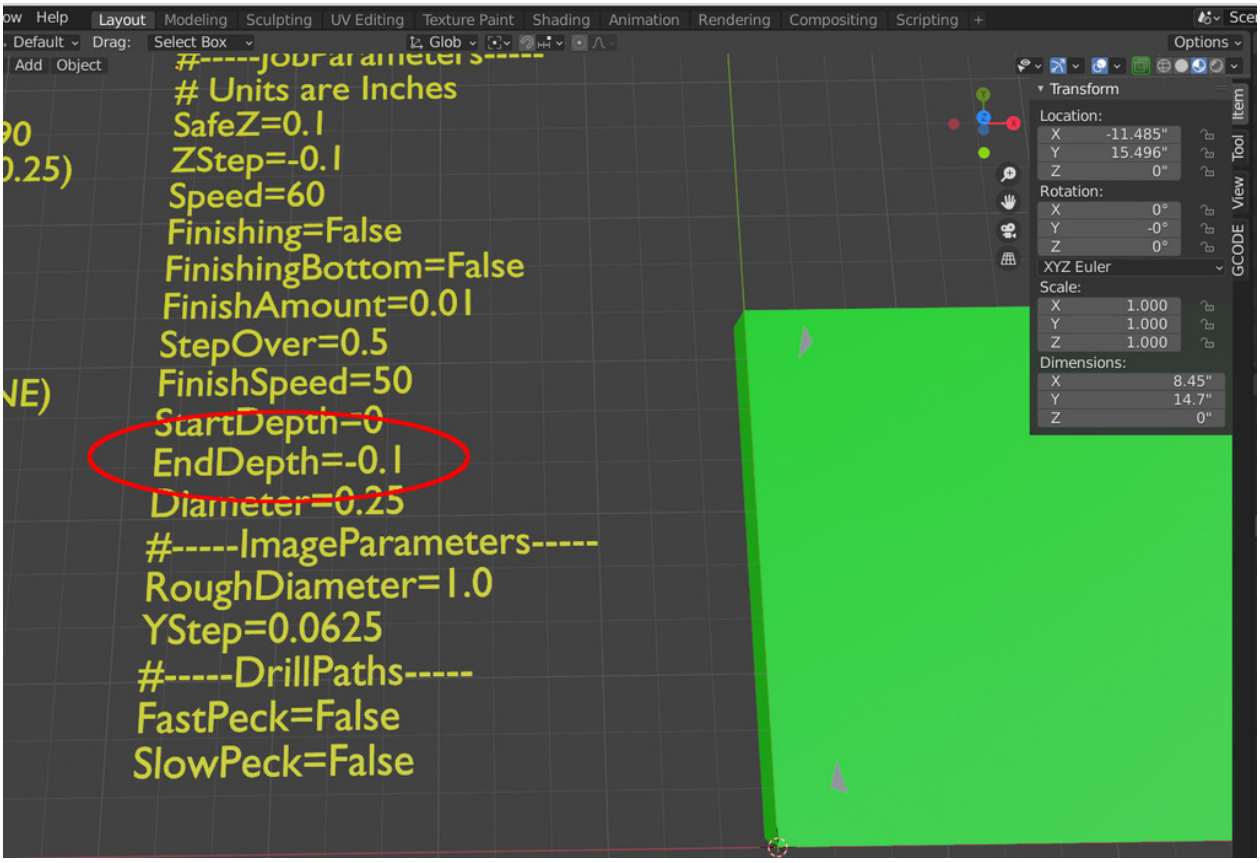

Figure 117: Because we know that most of the pocketing operations in this project are going to be set to a depth of 0.1, let's set the job parameters to have a default end depth of -0.1. Select (left-click) the JobParameters text object and press TAB to enter edit mode. Change the line as shown and then press TAB again to exit edit mode.

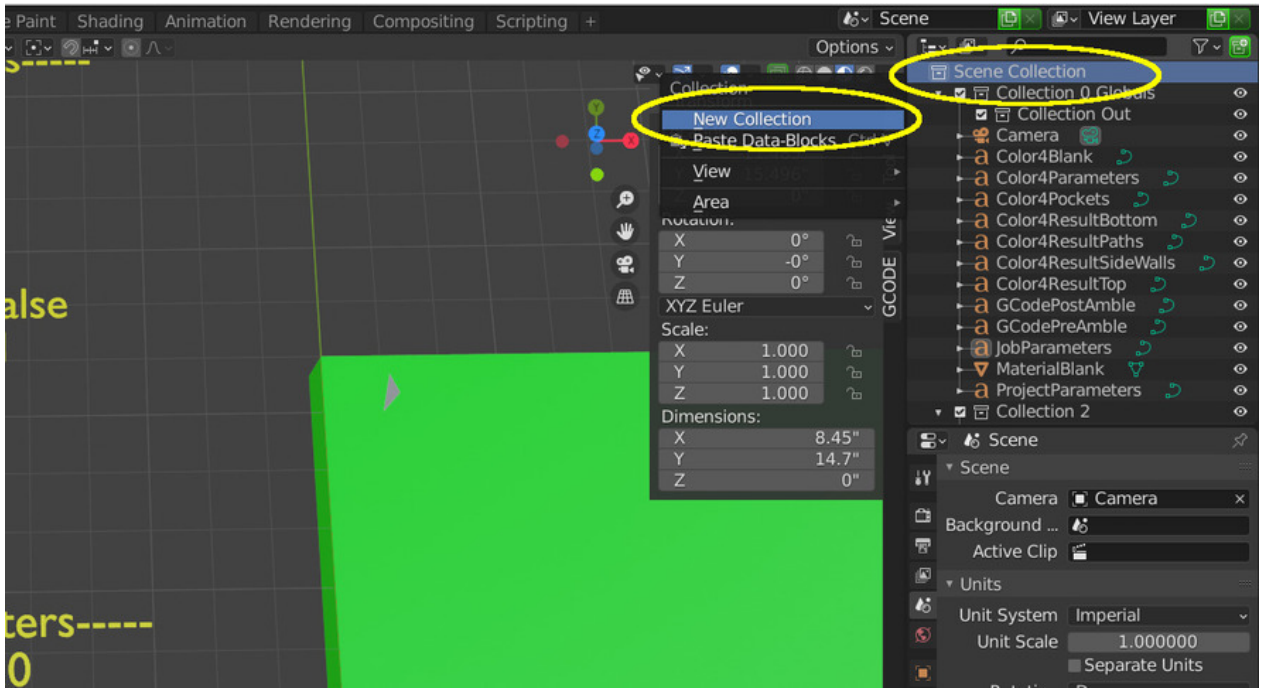

Figure 118: Because we know that this project is going to get complicated, let's organize the project with a few collections. Start by creating a new collection (we will move the 4 holes into this new collection). In the upper right of the display, right-click on "Scene Collections" and select "New Collection".

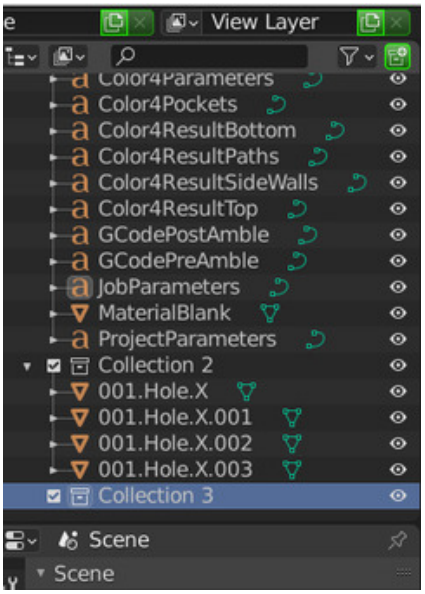

Figure 119: You should be able to see a new collection called "Collection 3" has been added (you may have to scroll down the list).

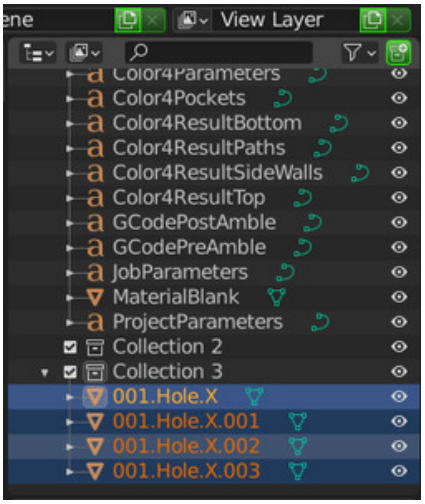

Figure 120: Select the 4 holes (if you do this in the list, you can just click the first hole in the list and then shift-click the last hole in the list). Then drag and drop the holes onto "Collection 3".

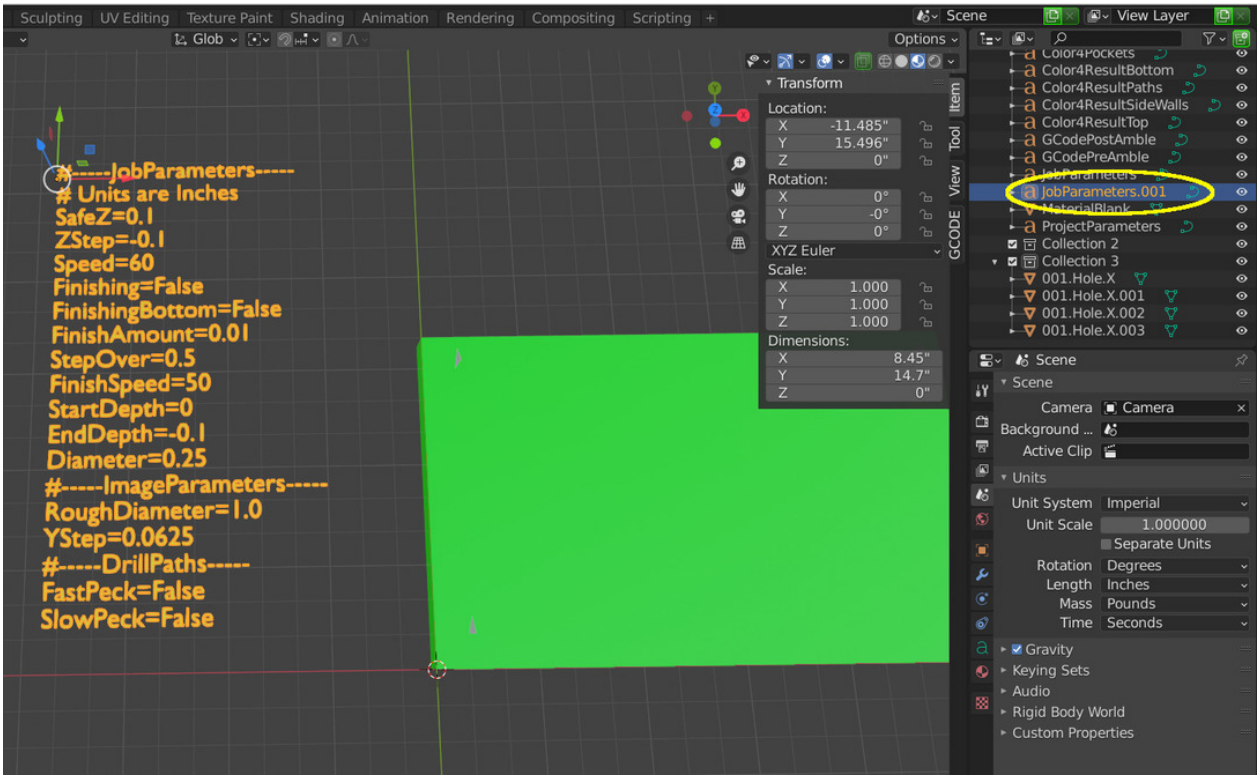

Figure 121: We are going to add a LayerParameters object to "Collection 3" so we can override some default values. We will do this by copying and renaming the JobParameters object. Select the JobParameters text object and duplicate it by pressing shift-d then Enter. You will see a new objext has been added to the list called "JobParameters.001".

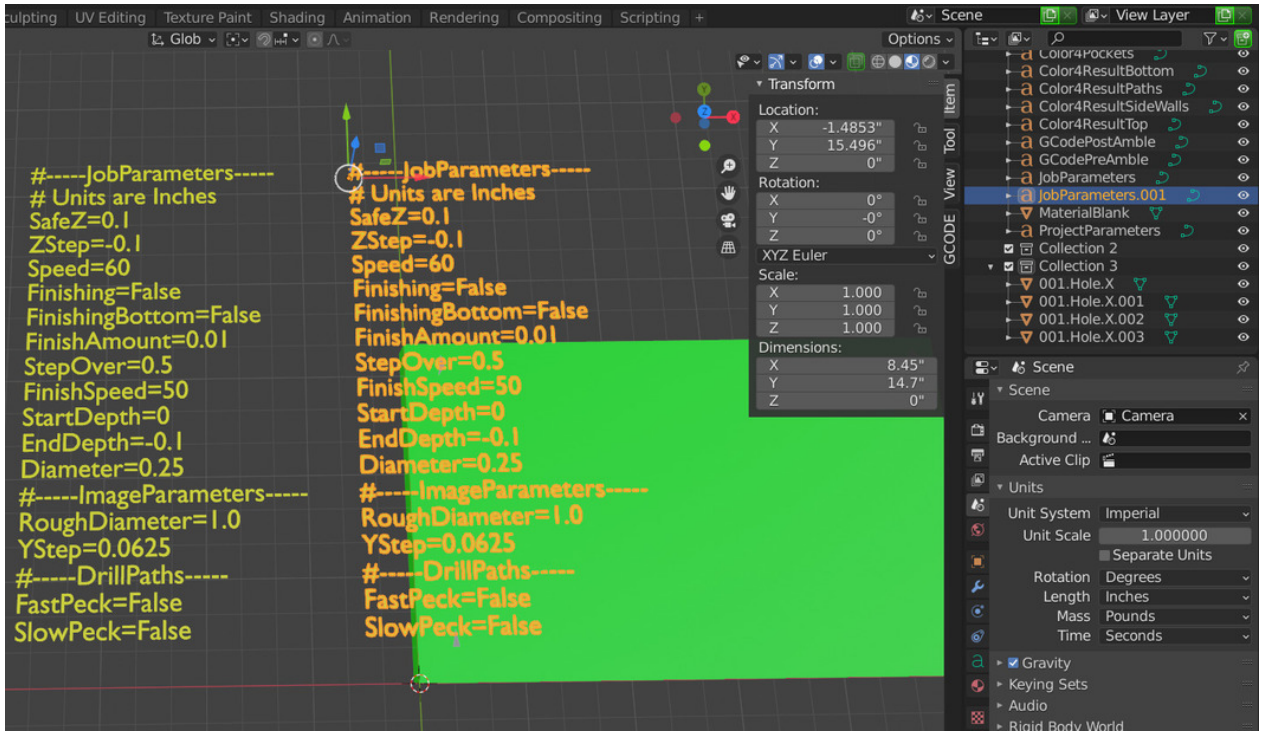

Figure 122: Move the new text object away from the original JobParameters object.

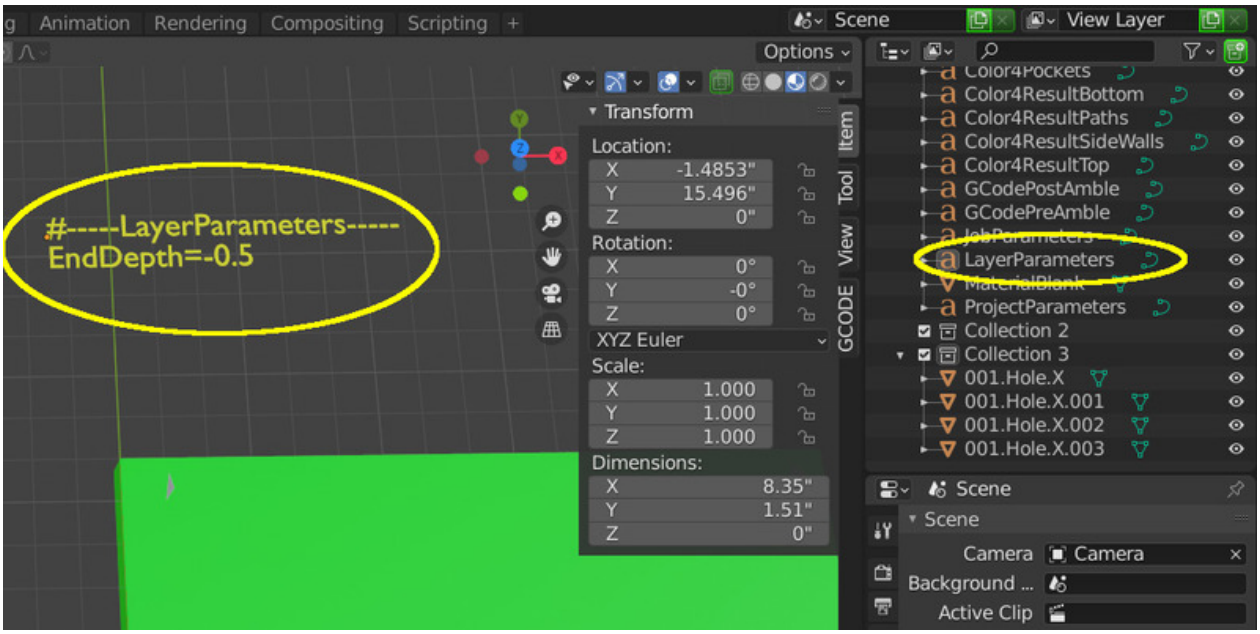

Figure 123: With the new parameters object selected, press TAB to enter edit mode and change the text to that shown in the image (setting the end depth to -0.5) and then press TAB again to exit edit mode. Then, in the list on the right-hand side, double click the "JobParameters.001" name and change it to "LayerParameters".

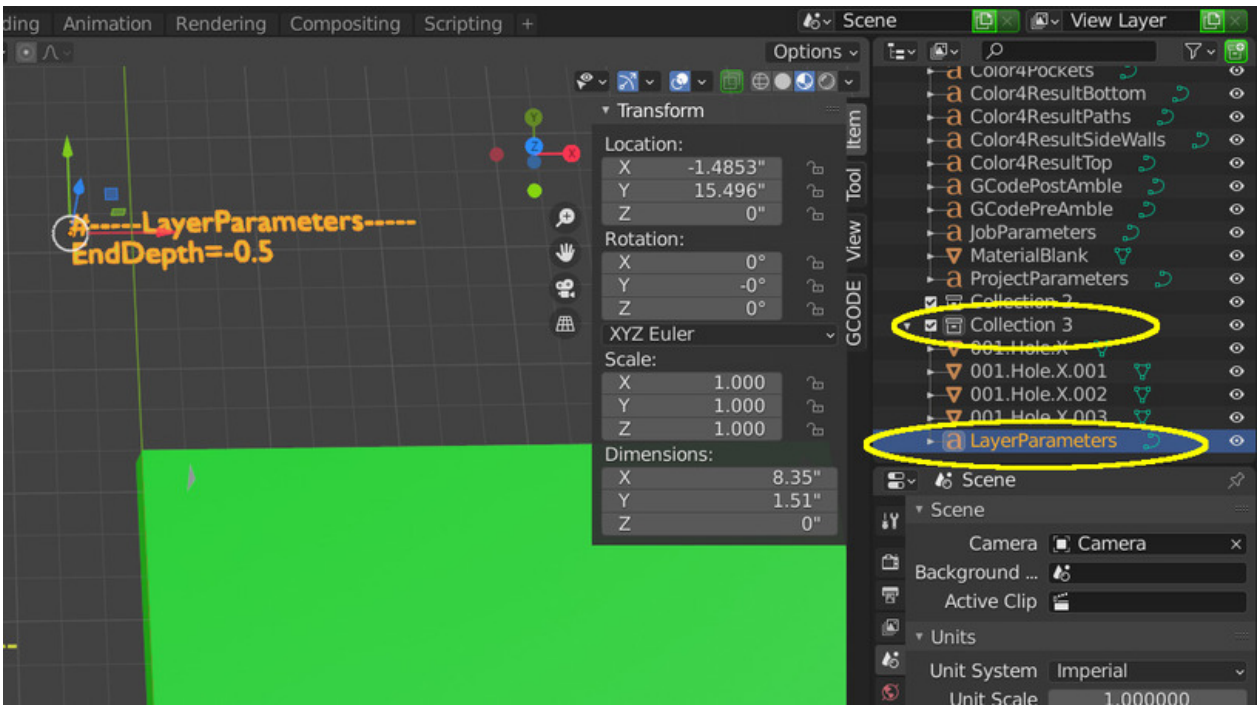

Figure 124: Add the LayerParameters object to "Collection 3" by dragging the name in the upper right section and dropping it onto "Collection 3". At this stage, it seems wasteful to have gone through the effort of adding this parameter object to "Collection 3" when we could have just left the default parameters set to -0.5 for the end depth, but there are more operations to add yet and the benefits will be shown later.

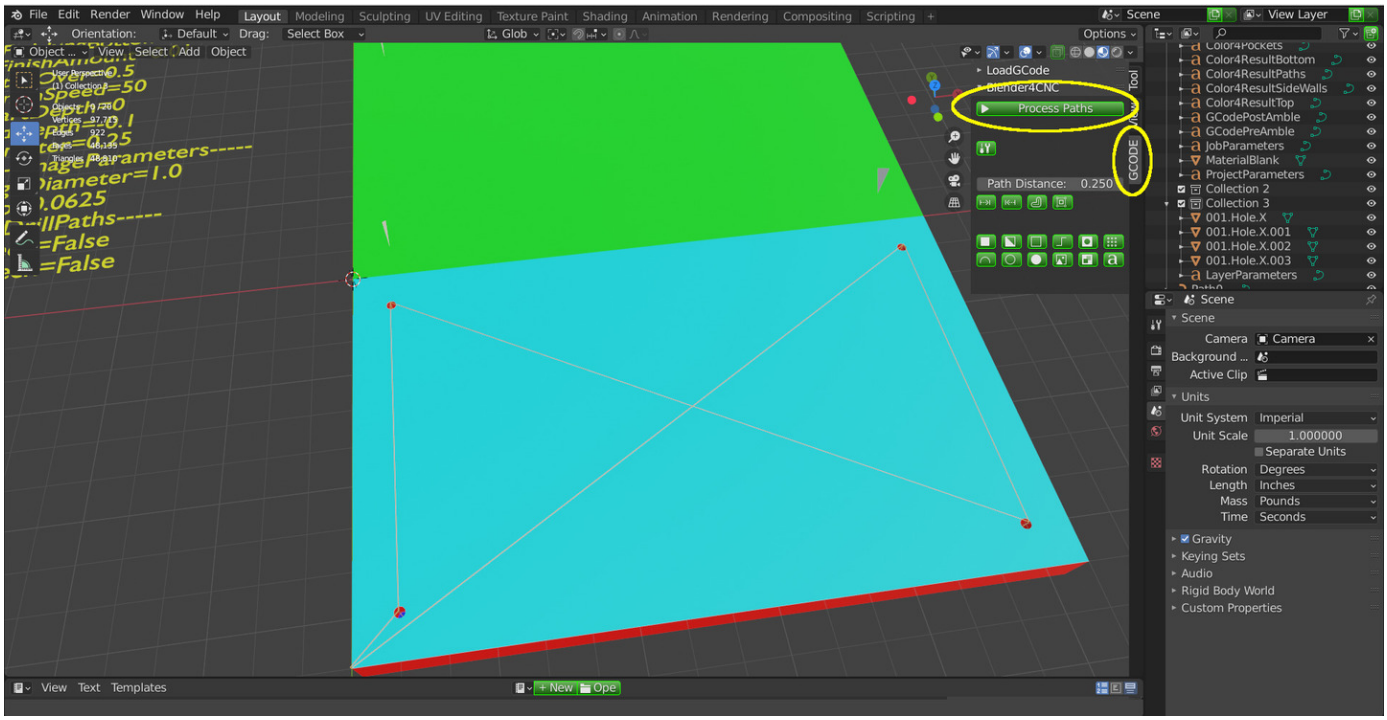

Figure 125: To see how we're doing so far, click the "Process Paths" button in the GCode tab. The visualization shows 4 holes being drilled (and they are being drilled to a depth of -0.5). If you get an error, it is likely that the hole objects were not added to a collection - see the next figure.

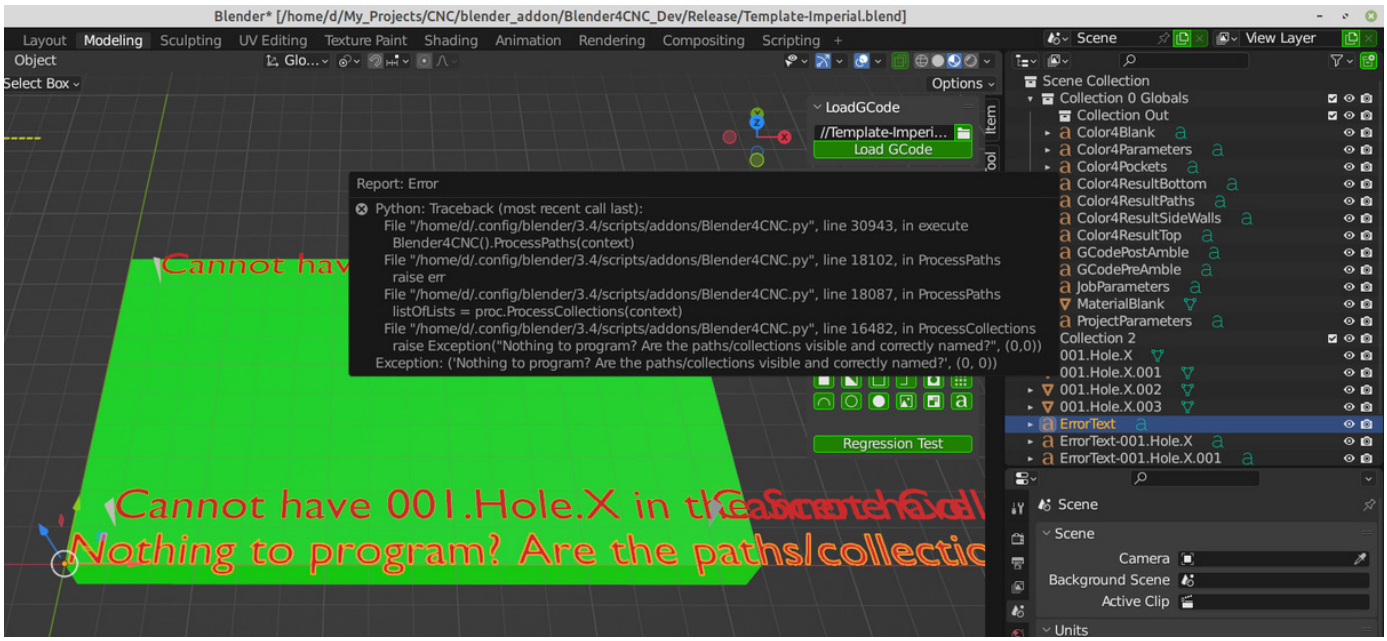

Figure 126: In this example, an error was returned when we clicked on "Process Paths". Read on to see how to correct this problem (it's easy).

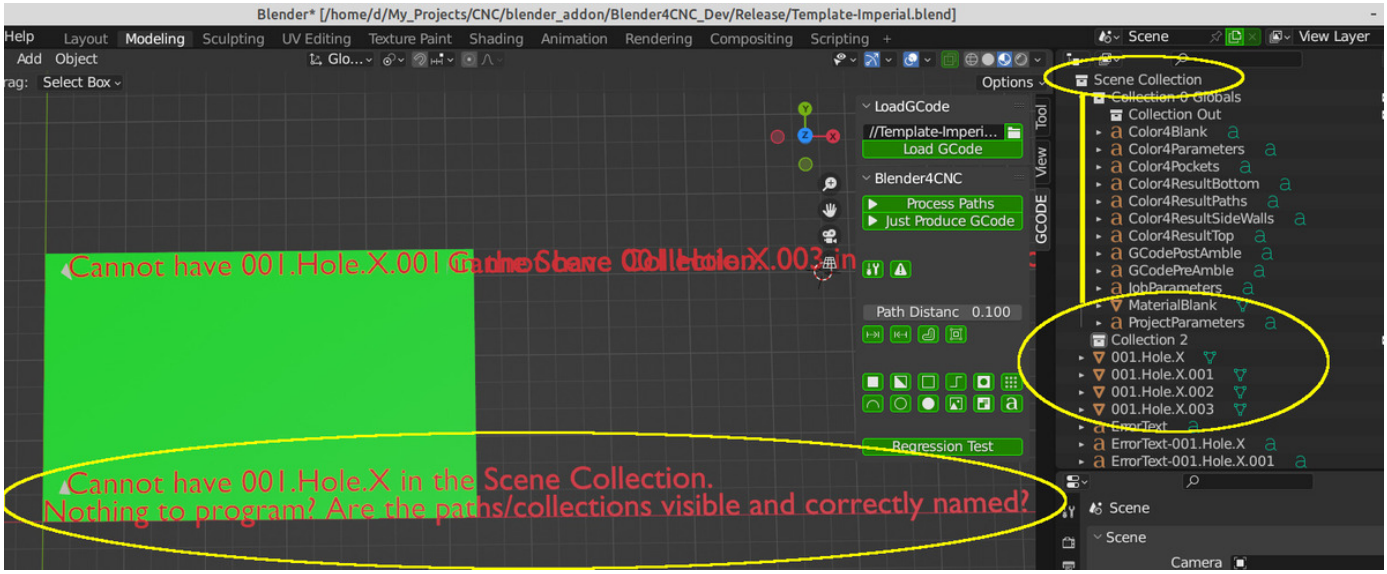

Figure 127: The error occurred because the hole objects are underneath the "Scene Collection" (which is the top-most collection) rather than being underneath and contained within the "Collection 2" collection. Note the error states that you cannot have the hole operation in the Scene Collection. Error messages that are specific to an operation appear near that operation and other job-level errors appear at the base of the material blank. The job-level error message is stating that it has found no valid operations to write GCode for (after it skipped the hole operations that were in the wrong collection, there were no other operations to process). Read on to fix this problem.

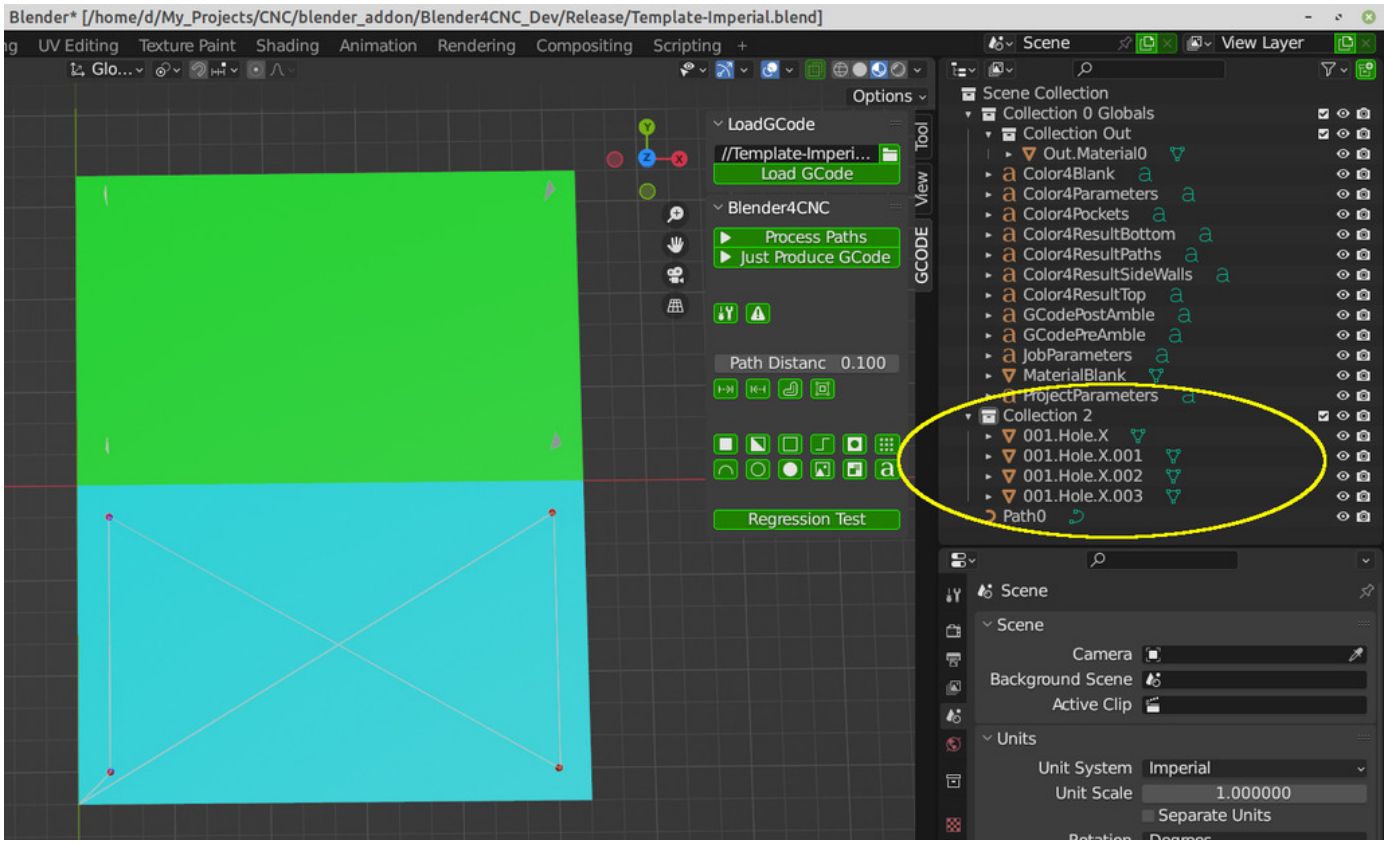

Figure 128: Simply select the hole operations in the upper right window and drag-and-drop them into the "Collection 2" collection - you should now see that they are indented underneath the collection. Then click "Process Paths" again. Remember, all operations must be contained within a collection (not the main "Scene Collection")!

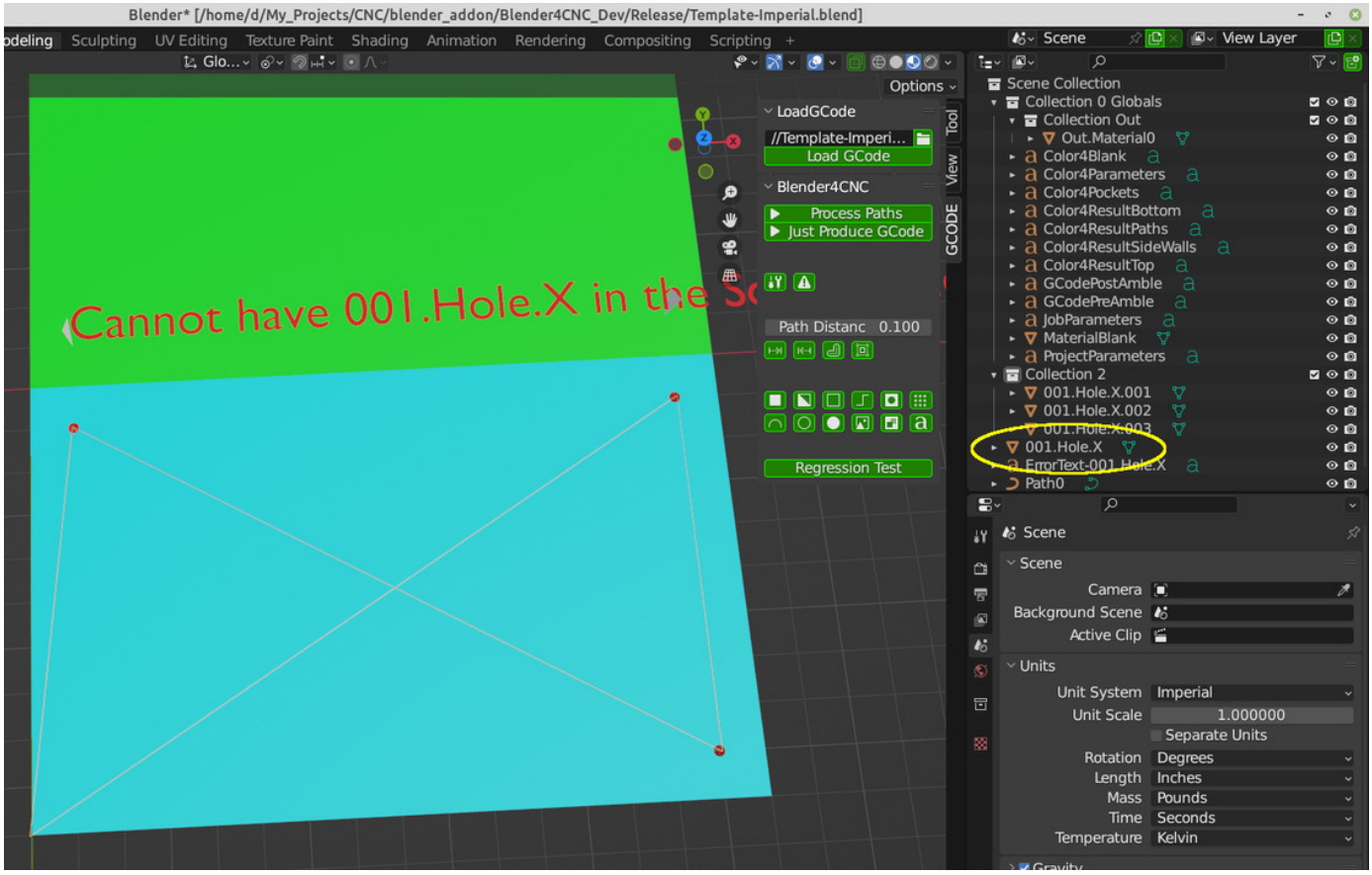

Figure 129: Here is an example of what happens when some operations are in a valid collection and some are not. Clicking "Process Paths" will still render all valid operations (in this case, 3 holes) but will place an error message beside any operation that is not in a collection (in this case, the first hole was moved out of the collection and it was not processed and an error shown).

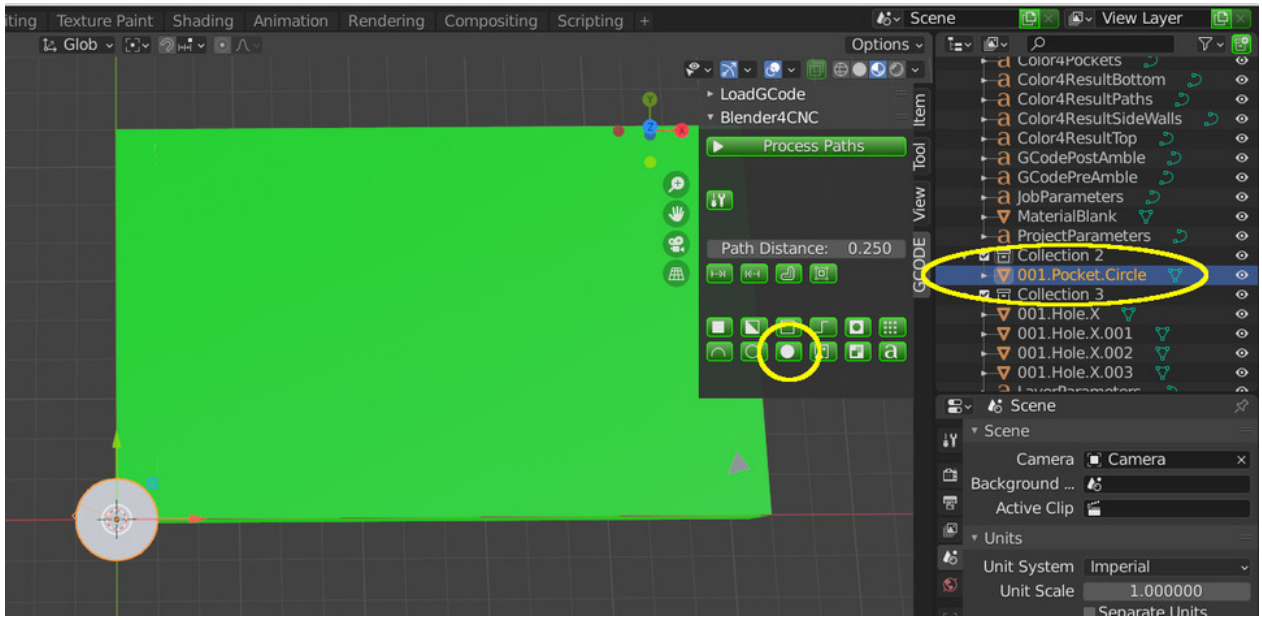

Figure 130: Don't forget to save your progess regularly, just make sure you give the project a unique name when saving so you don't overwrite the "Template-Imperial" project. Click on "Collection 2" to make certain that new operations are added into that collection. (If at any time you create operations and they do not appear in the collection you assumed they would, you can just drag them into the desired collection.) Then click on the "Create Circular Pocket" button and you should see a new operation appear called "001.Pocket.Circle".

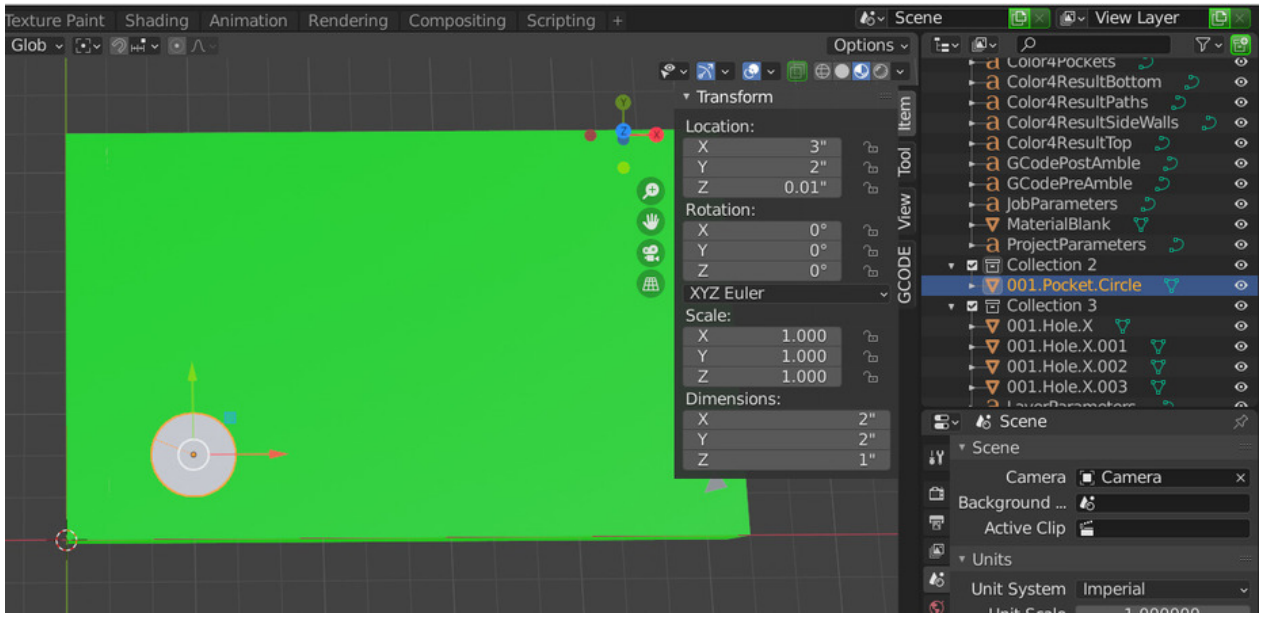

Figure 131: Move the new circular pocket to coordinates X,Y = (3,2). Either drag it or set it via the item tab.

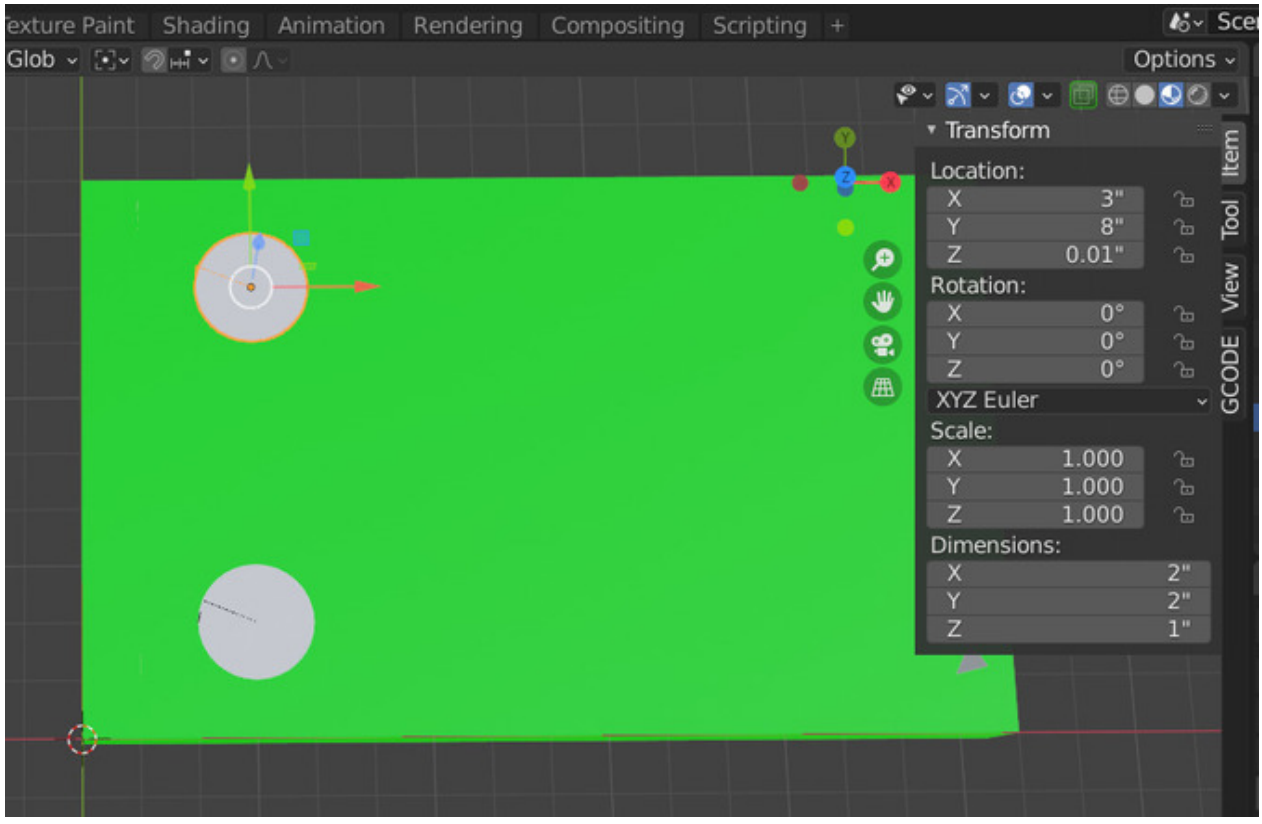

Figure 132: Duplicate the circlular pocket (shift-d, then Enter). Then drag the new circle to coordinates X,Y = (3,8).

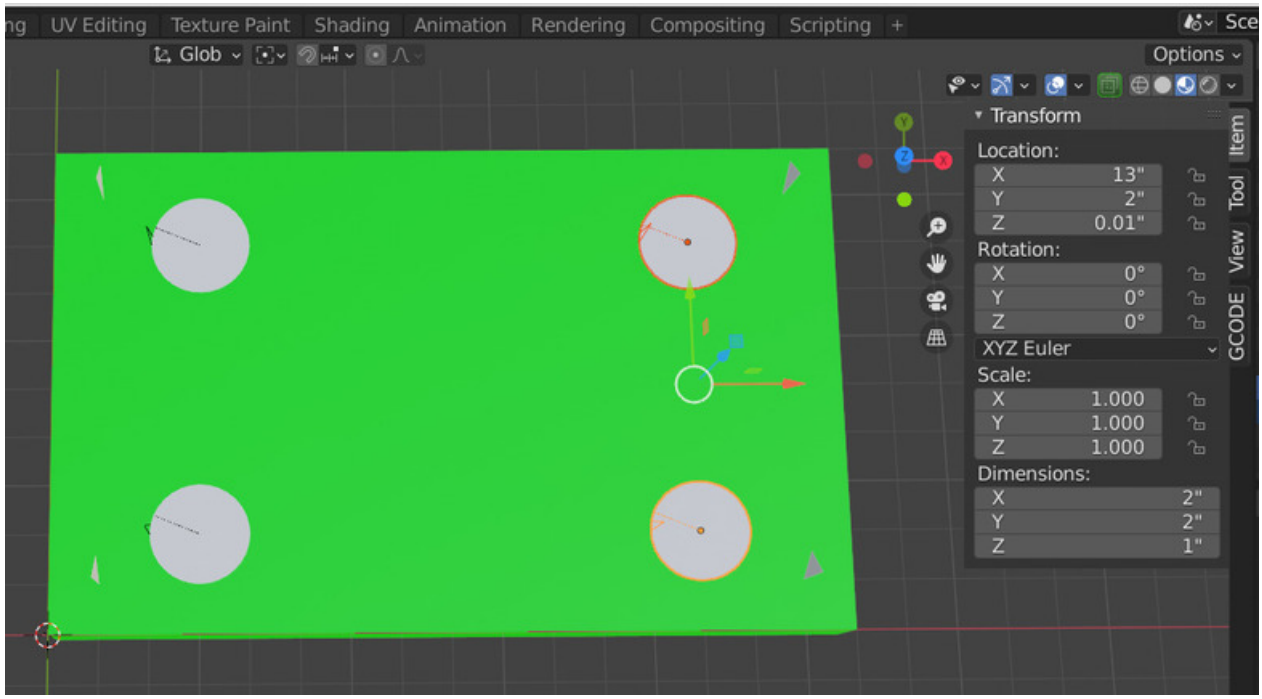

Figure 133: Now select both circular pockets (click one pocket then shift-click the other pocket) and copy them both (shift-d, then Enter). Then drag the new pair of circular pockets over until they are both at X = 13. If you prefer not to drag the operations and instead prefer to set locations via the items tab, (maybe your grid is not set to 1" intervals for example) you may notice that when you have multiple items selected and you type into the X,Y or Z field of the item tab, only **one** of the objects is moved. Blender **does** have a way of applying a number to all selected objects - in this case, with two circular pockets pockets selected, go to the items tab and enter X=13. You will see that only one of the circular pockets has moved. Now, right click on the "X" field that you just set and in the pop-up menu, select "Copy Single to Selected" - you should see the other circular pocket jump to X=13 also. Either use this trick when settings common locations to multiple objects or just take a little more time to do each one at a time.

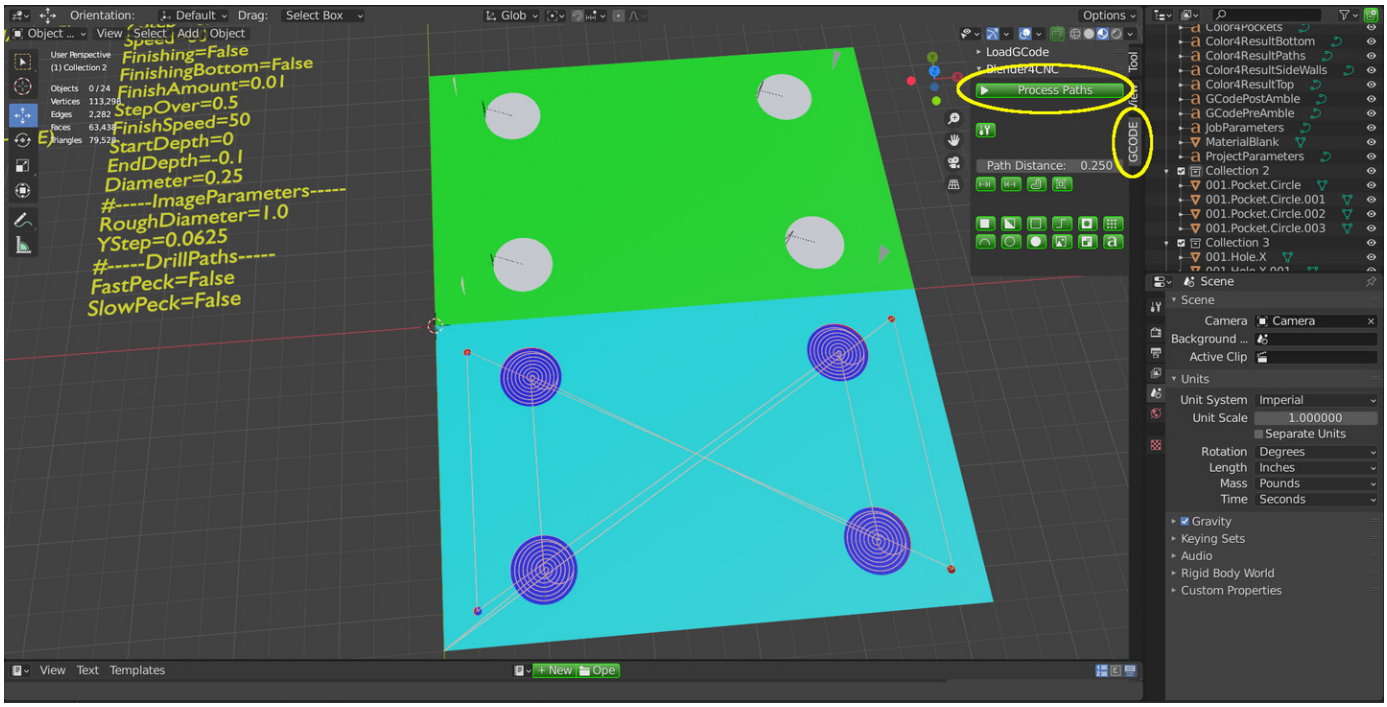

Figure 134: To see how we're doing so far, click the "Process Paths" button in the GCode tab. You should see 4 drilled holes and 4 larger circular pockets..

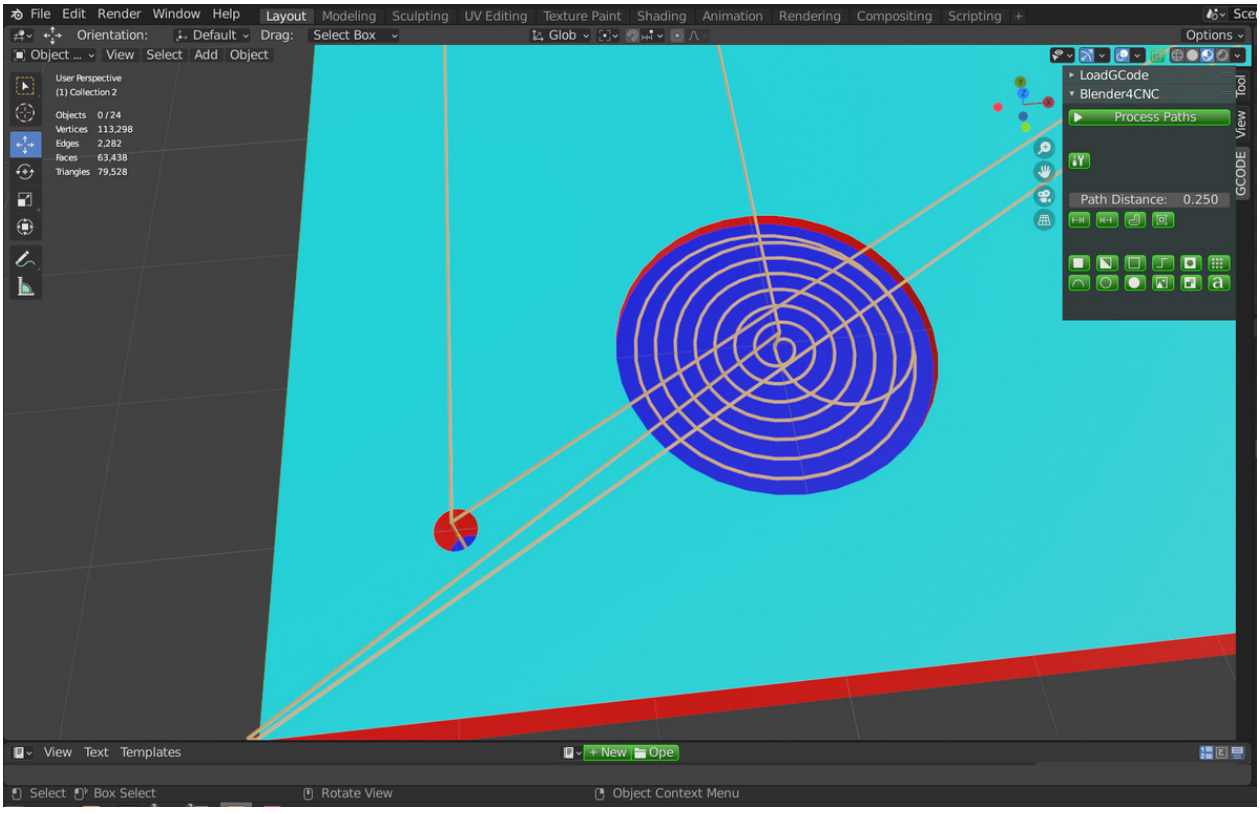

Figure 135: By zooming in (mouse wheel or middle mouse button) to the results and tilting the view (holding down both mouse buttons while moving) you can see that the larger circular pockets are being cut to the default depth of -0.1 and the holes are being drilled to -0.5. You can also see the path showing how the cutter path is spiraling around to cut out the material inside the pockets.

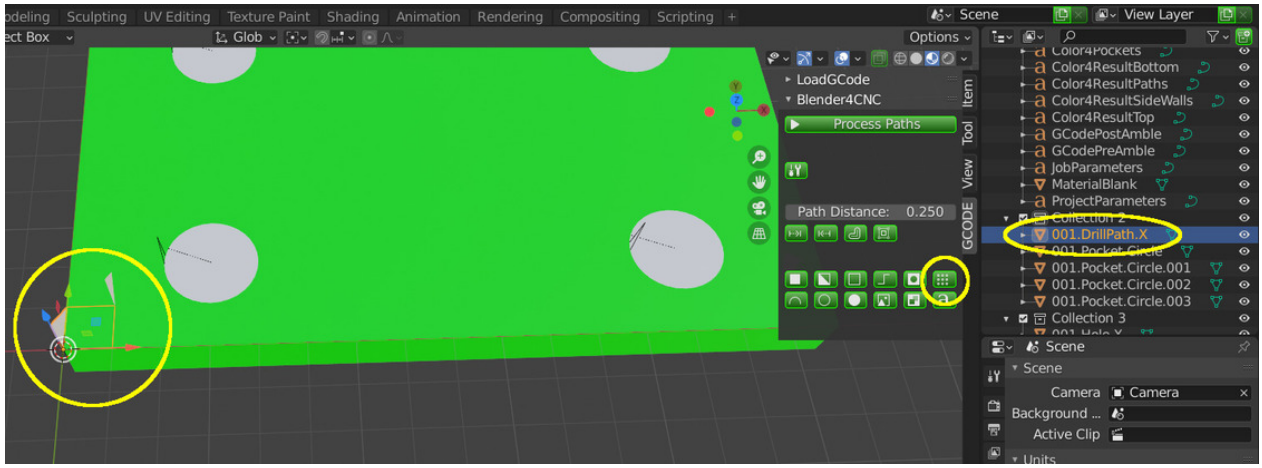

Figure 136: Let's add two rows of dots or holes. Click the "Create DrillPath" button and you will see a drill path operation appear called "001.DrillPath.X".

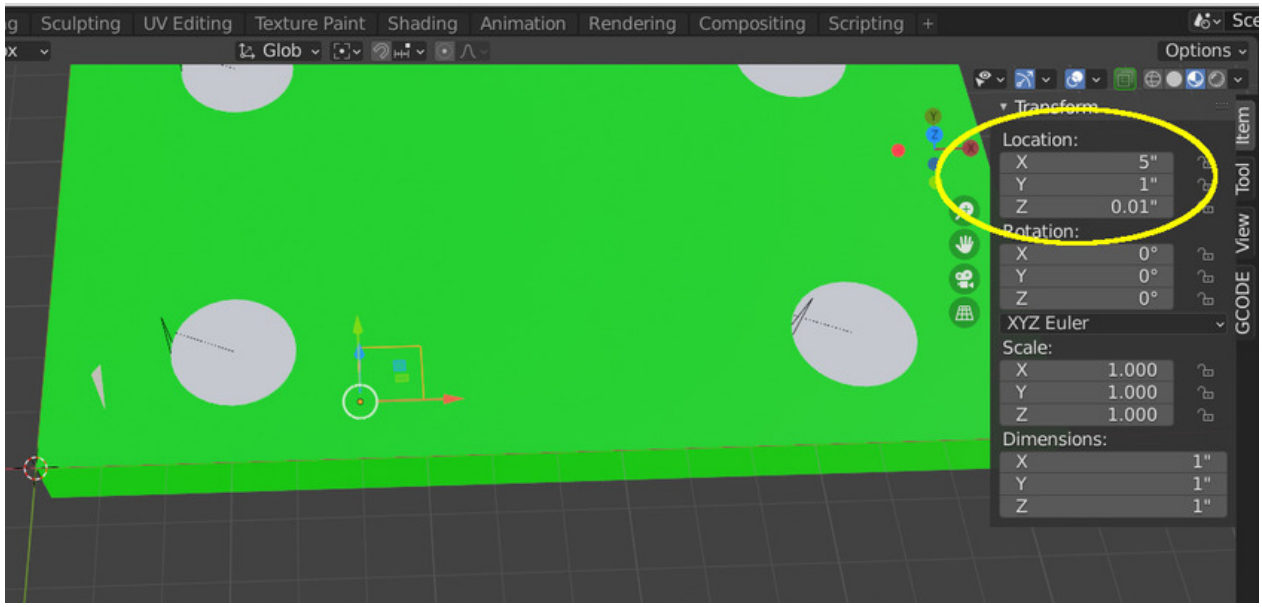

Figure 137: Move the drill path to location X,Y = (5,1).

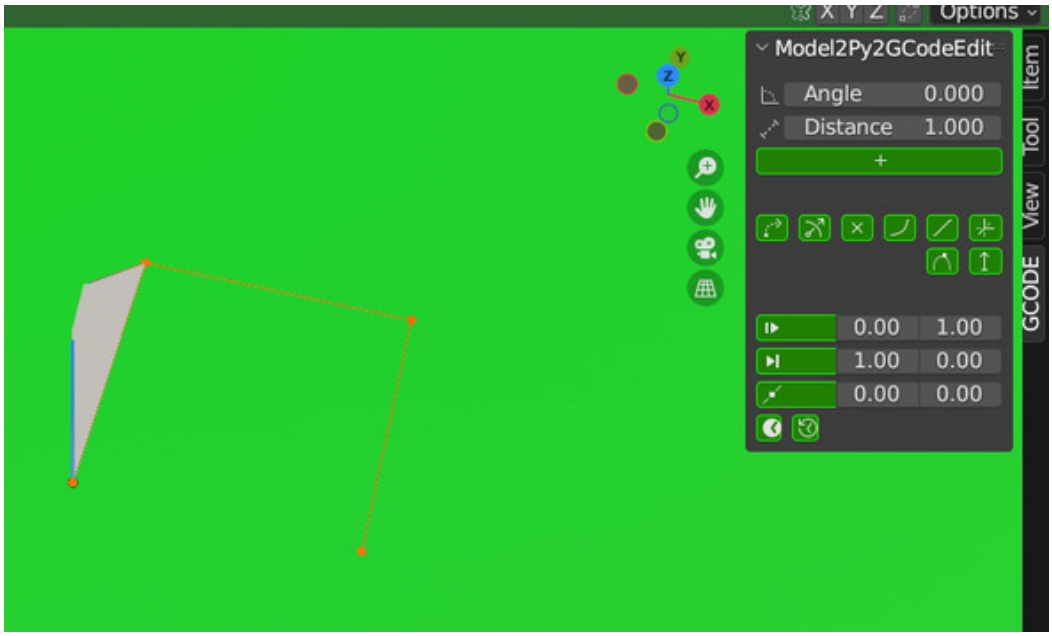

Figure 138: We are going to edit the drill path to add more points (more holes) along the path. Zoom in a little and then press TAB to enter edit mode. Initially, all the points will be selected (shown in orange) - just click somewhere away from the points and they will all become unselected (and show in black).

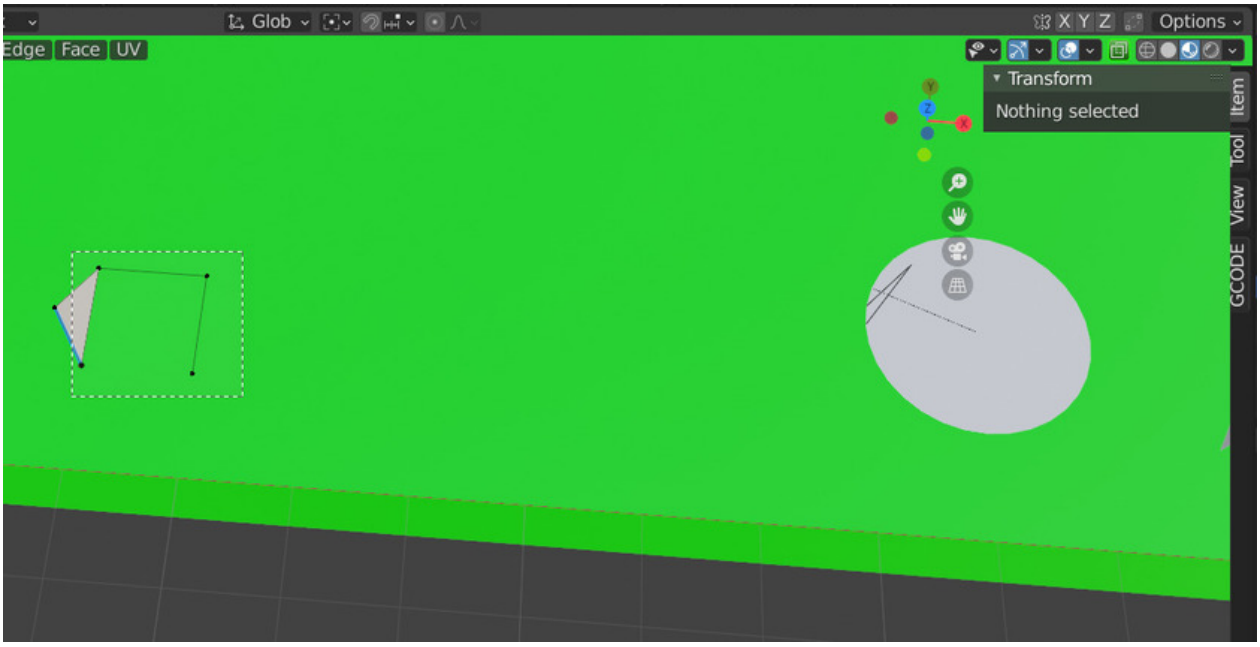

Figure 139: Select the four main points (not the top of the "start" point). You can left-click with the mouse and drag a selection rectangle or you can click one point and then shift-click the remaining 3 points.

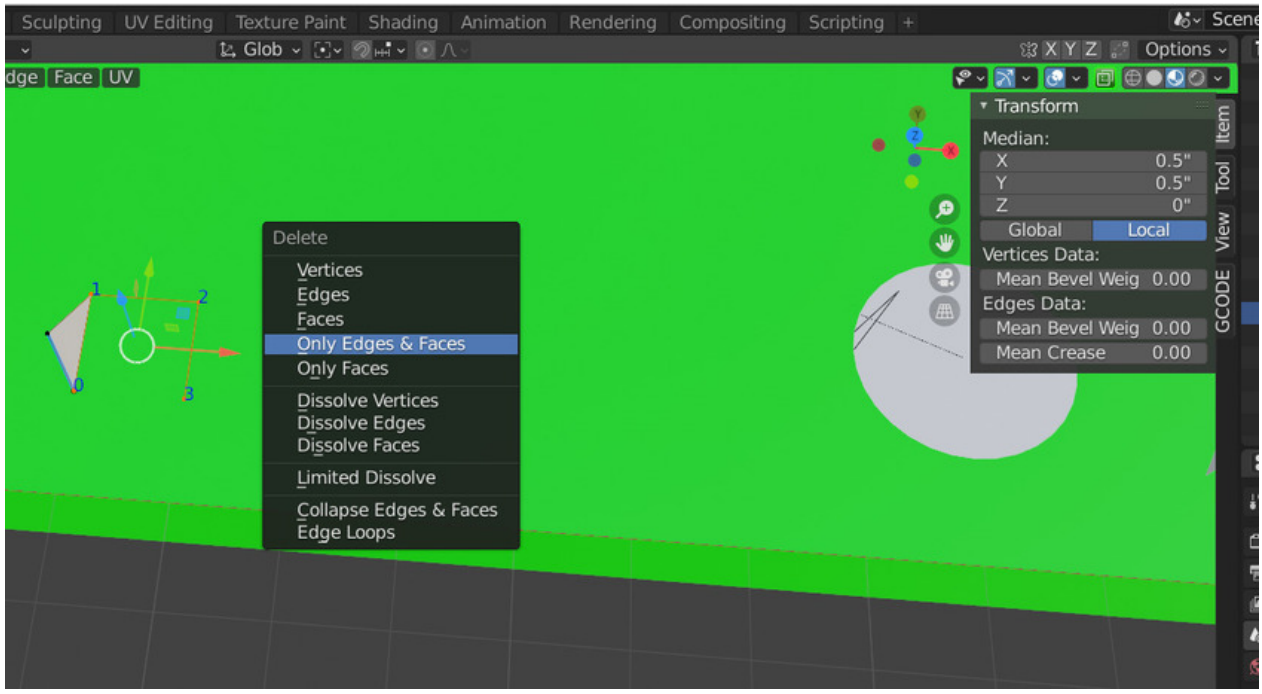

Figure 140: Press the "Delete" key and then select "Only Edges and Faces" from the popup menu. This will delete edges connecting the vertices but will NOT delete the vertices themselves. Had we have chosen "Edges" from the popup menu then any vertices that were not connected to unselected points would ALSO have been deleted as part of each edge. We want to keep the vertices.

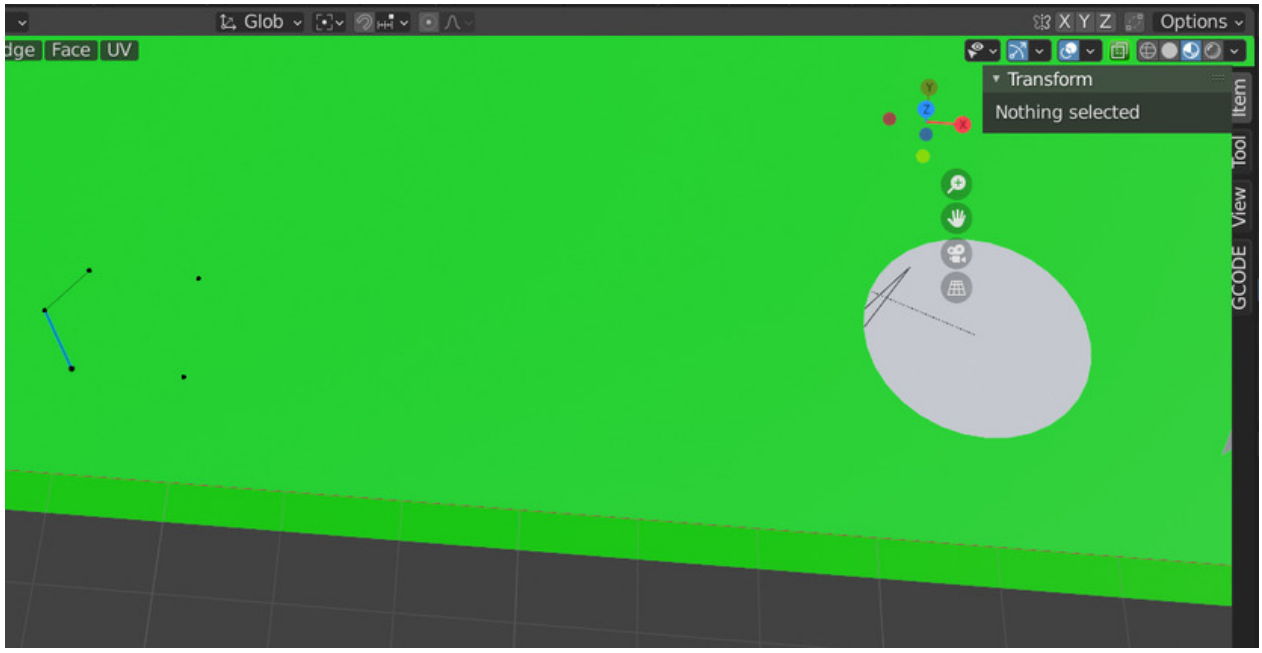

Figure 141: Only two edges remain.

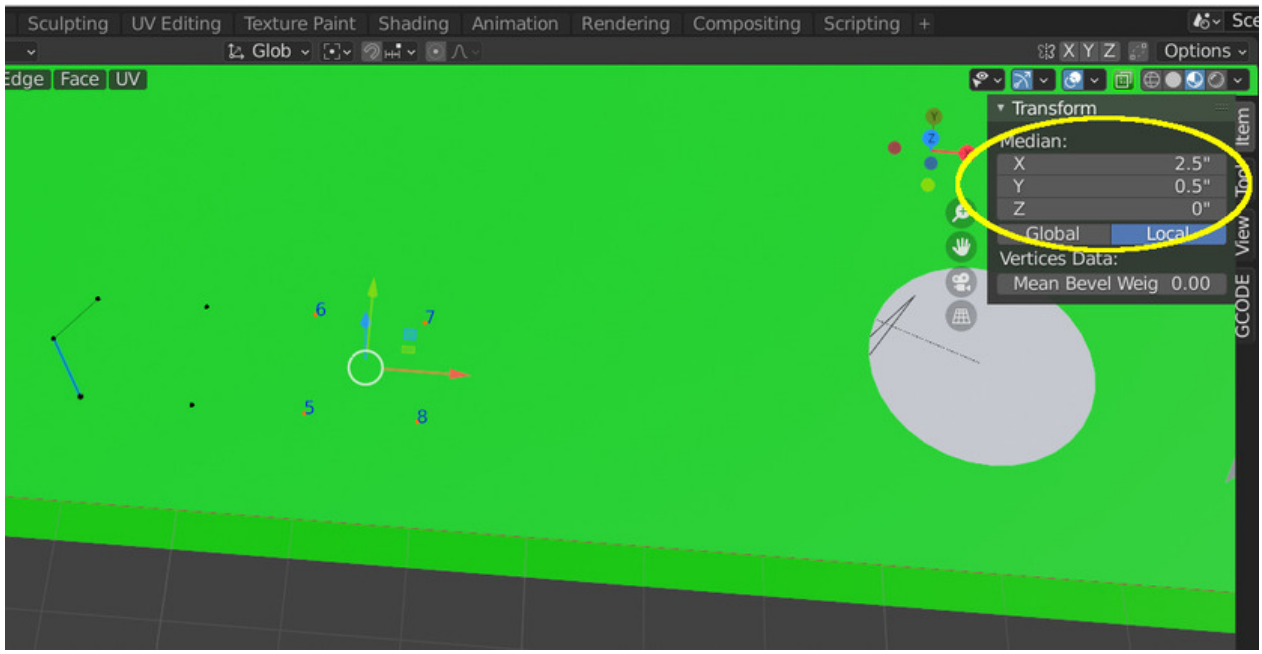

Figure 142: Select the 4 main points again and duplicate them (shift-d, then Enter). Then drag them (hold ctrl to snap to the Blender grid helps) to the shown position. (When you become proficient in Blender, you can press "g" and then "x" to move the 4 new selected points along the X axis rather than clicking and dragging the arrow. When you are really proficient, you can duplicate and move with the sequence - "shift-d", "x", then hold CTRL while moving the mouse to snap to a location. Or without using the mouse at all - "shift-d", "x", "+", "2", "ENTER" will duplicate and move +2 inches along the X axis.)

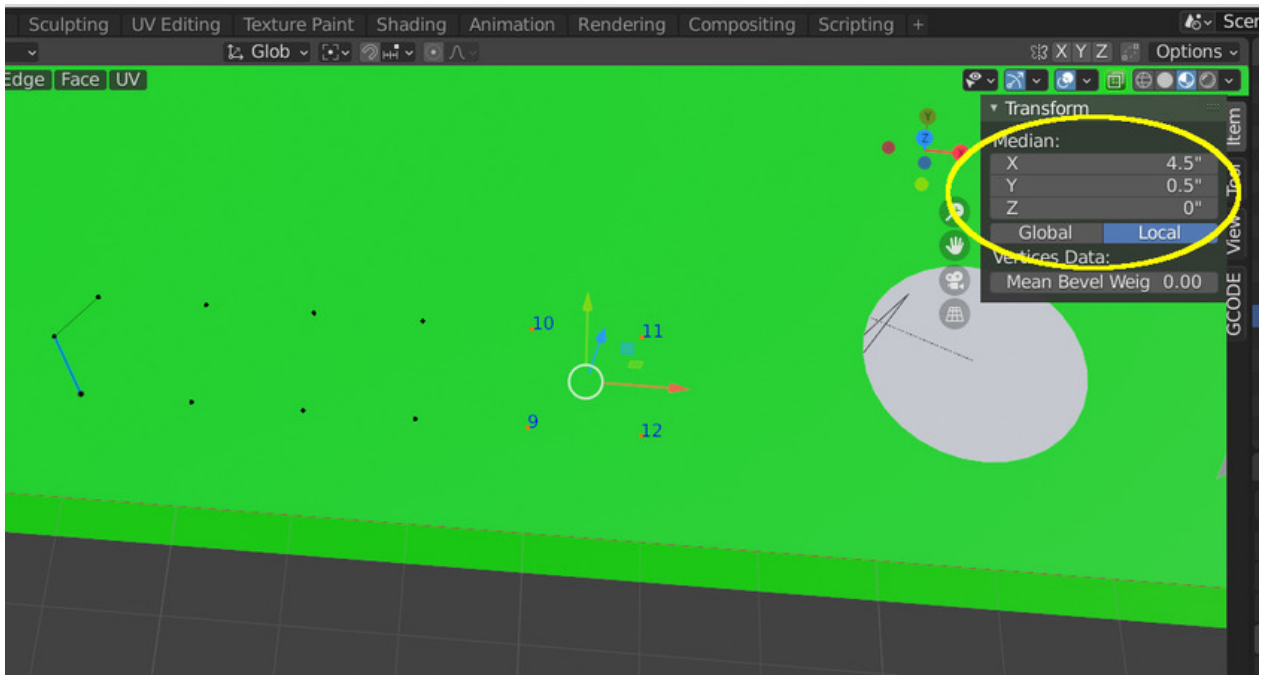

Figure 143: The 4 newly duplicated points should still be selected (if not, select them) and duplicate them again and move them as shown.

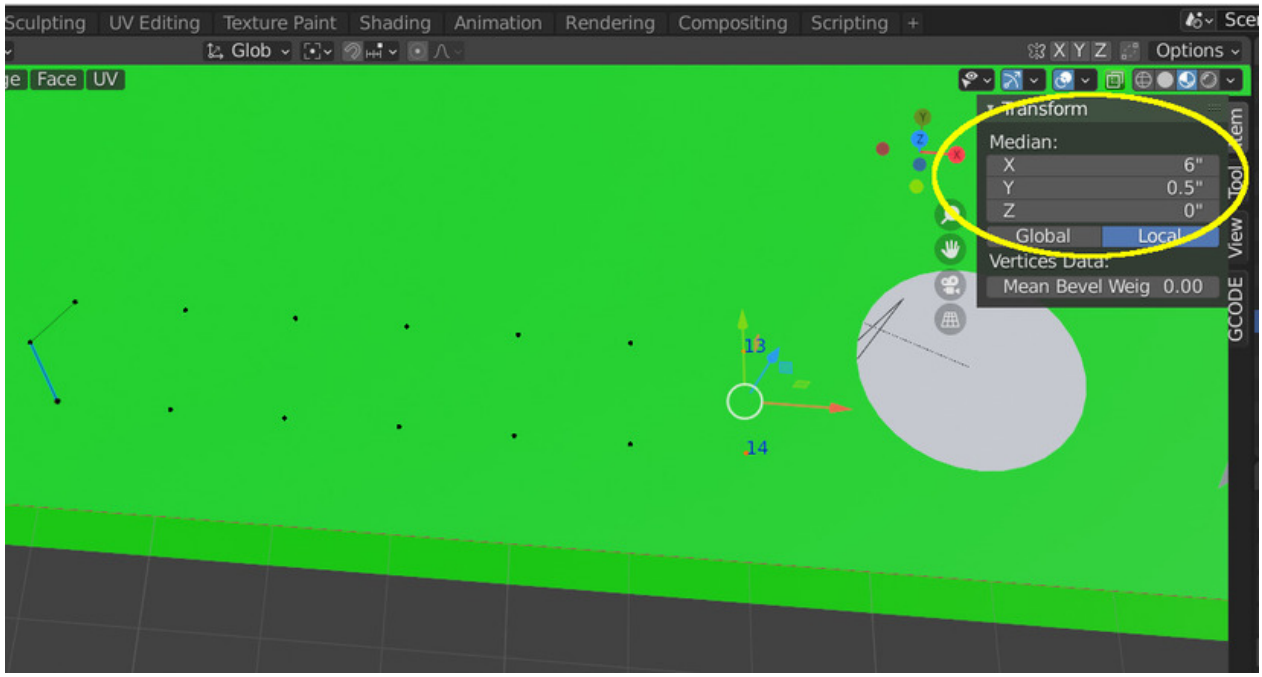

Figure 144: We only want the drill path to contain 7x2 holes so just select the last two points and duplicate and move them.

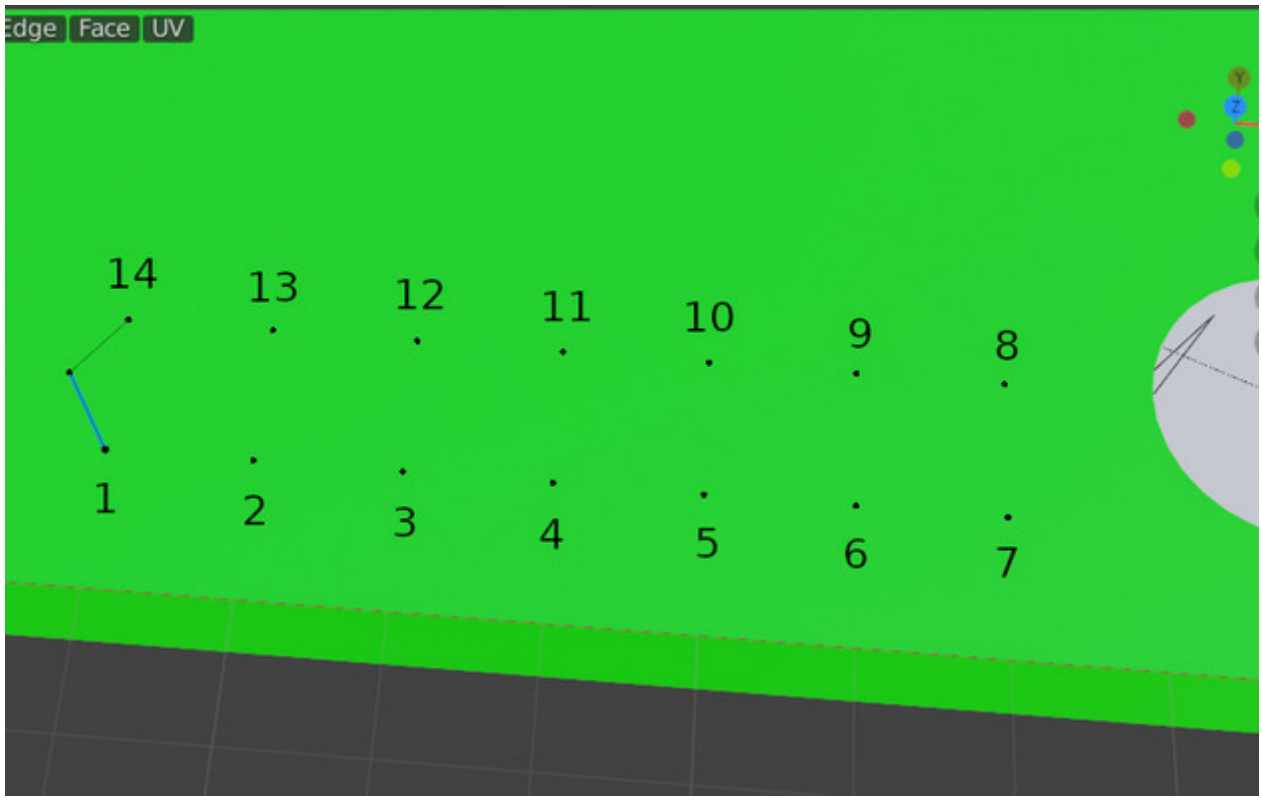

Figure 145: Now that we have the vertices where we want holes drilled, we must join them together into a path. We want to make the path efficient. The way to connect two points with an edge is to select two points and then press "f" to create an edge between them - we would have to do this 13 times. (In Blender, the "f" key means create a "face" or an "edge" if only 2 points are selected.) There is a quicker way if we are careful in the order in which we select the points. Select the first point (labelled "1") and then shift-click the remaining 13 points in the order shown (2, 3...) until they are all selected. Then press "j" (j stands for "join") and the points will be connected intelligently from one to the next.

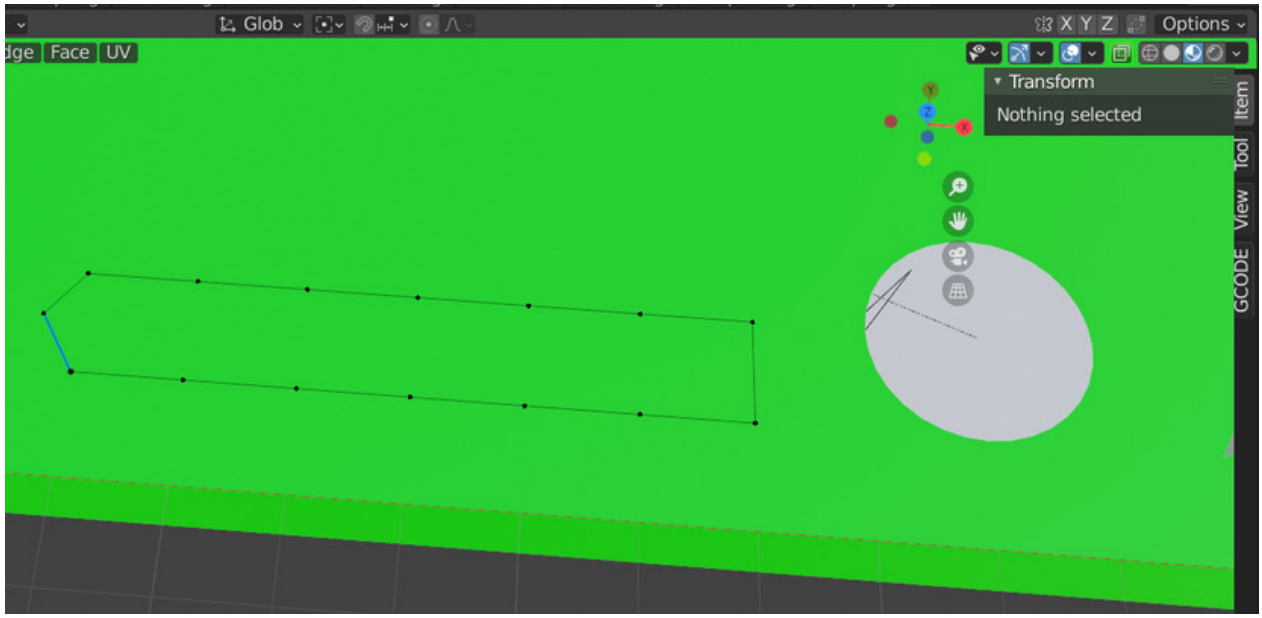

Figure 146: Click somewhere (not on a point) so that all the points are unselected and you can see how the path looks clearly. We are nearly done. We just have to fix the "start" flag.

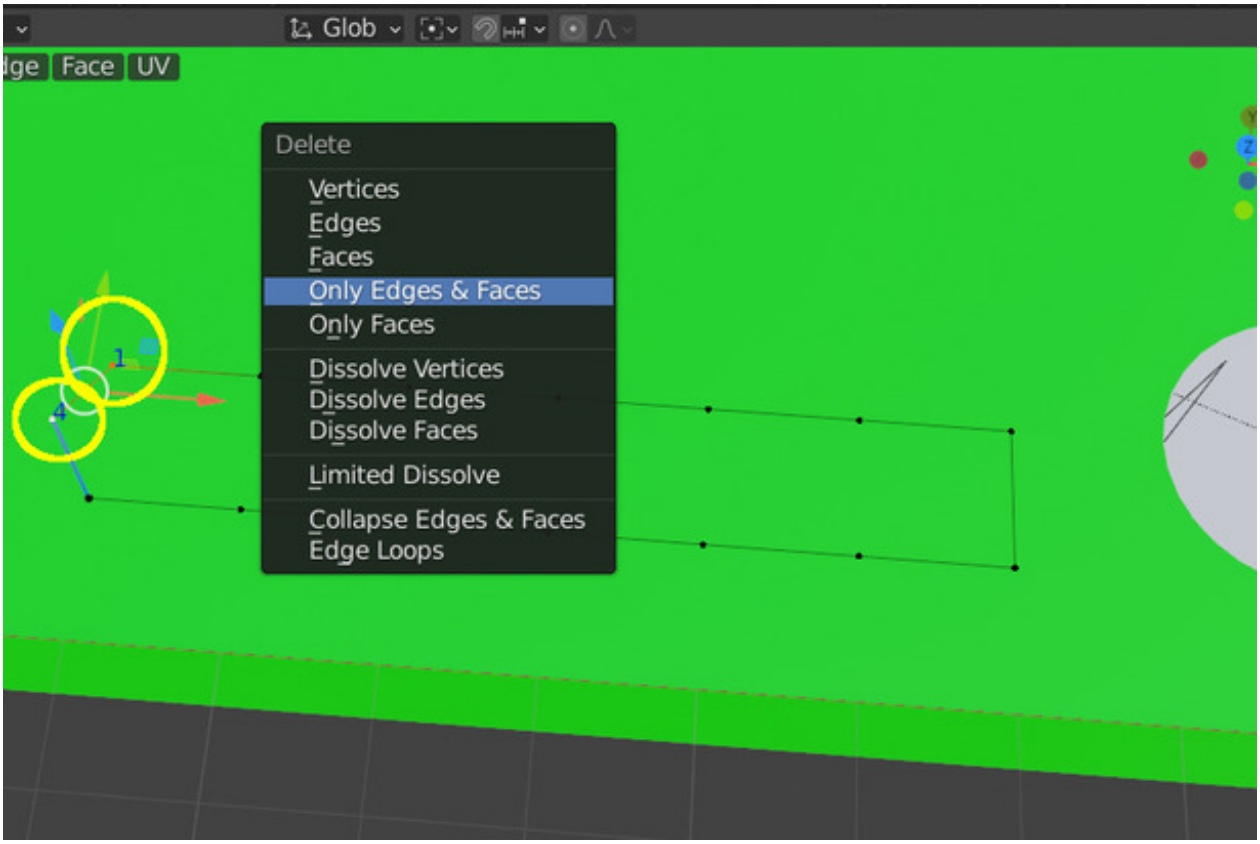

Figure 147: Let's delete the "start direction" edge. Select the point at the "top" of the blue start line and the last point (either draw a selection rectangle around these two points or click and shift-click the two points). Press "Delete" and select "Only Edges and Faces".

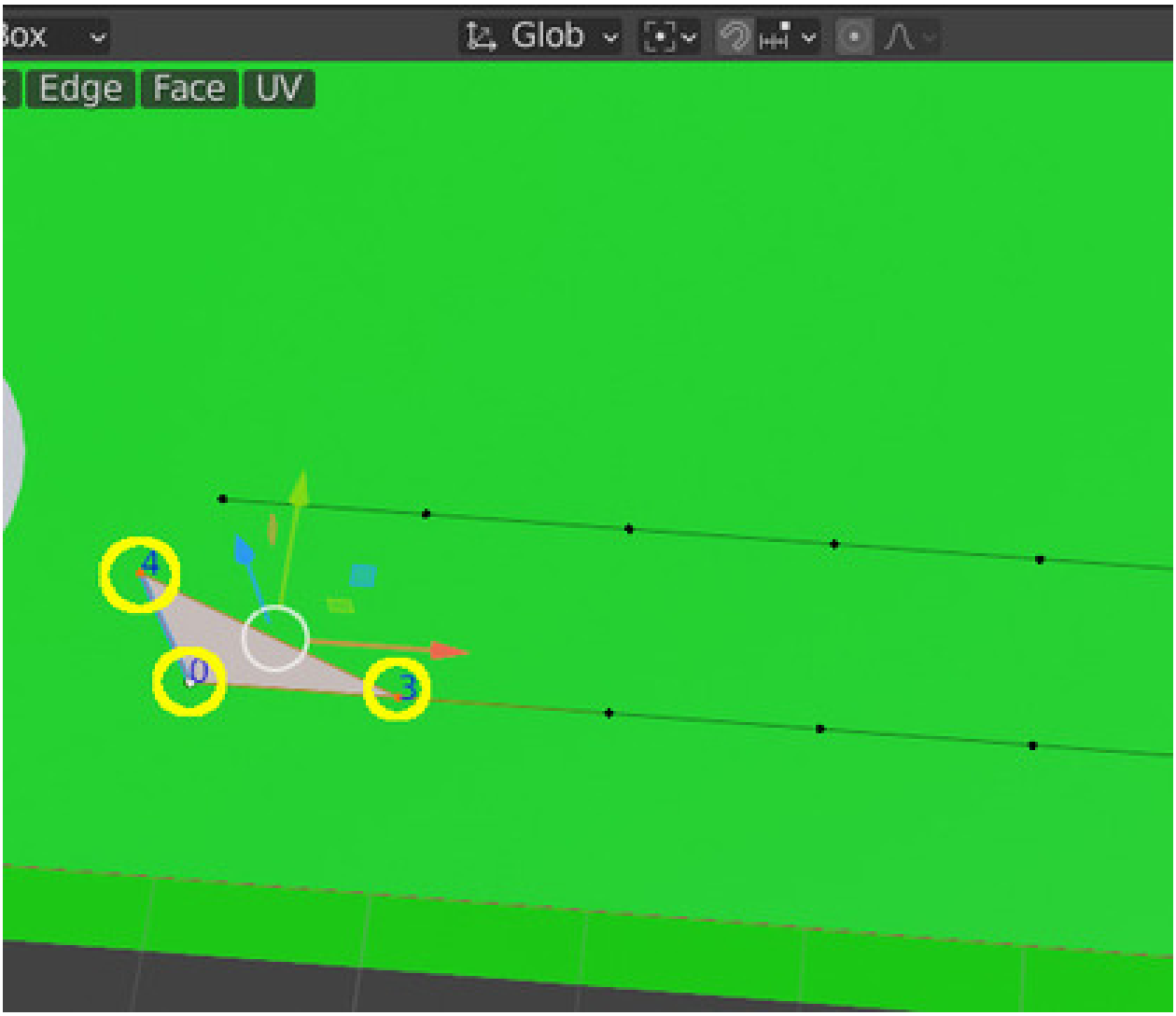

Figure 148: Create a new "start direction" flag. Select 3 points - the point at the top of the blue line, the 1st point and the 2nd point. Then press "f" ("face").

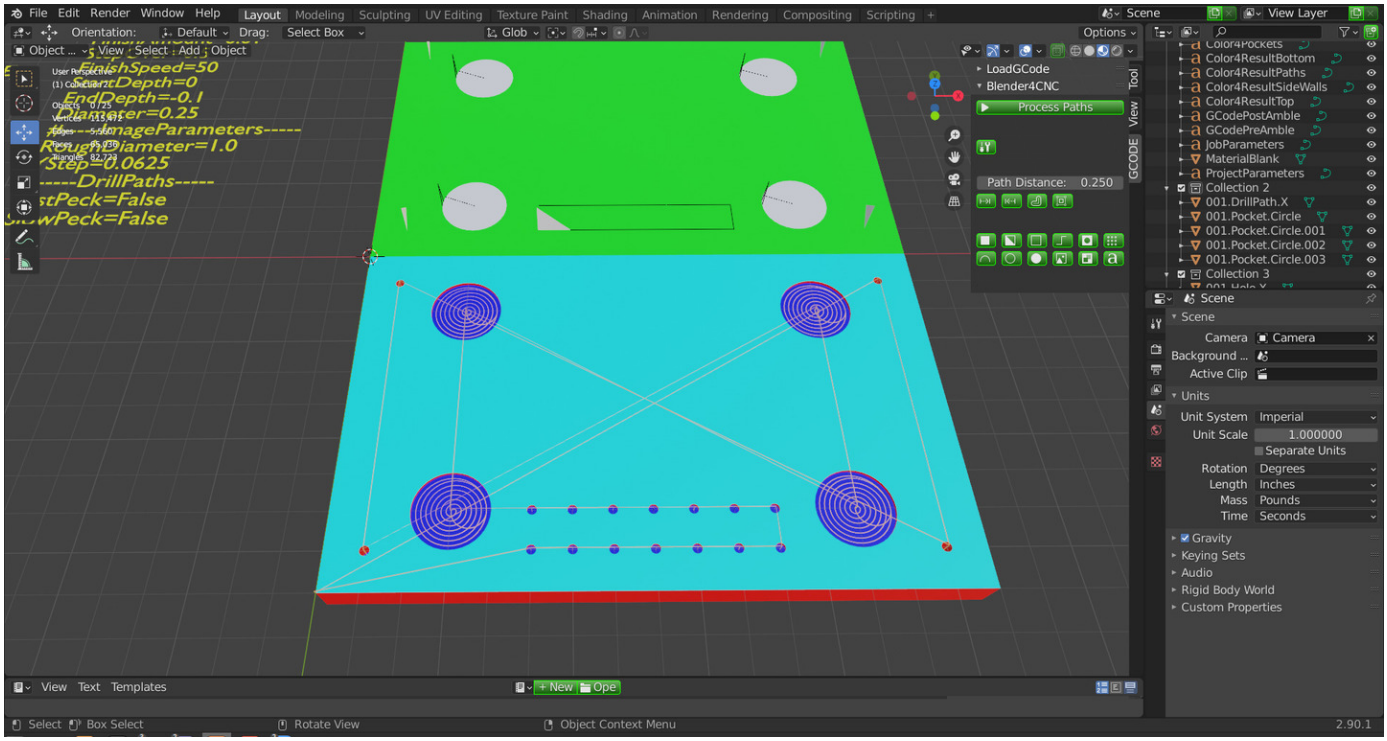

Figure 149: Press TAB to exit edit mode (should see "Object" in the upper left of the screen instead of "Edit") and then click "Process Paths" to see the progress so far.

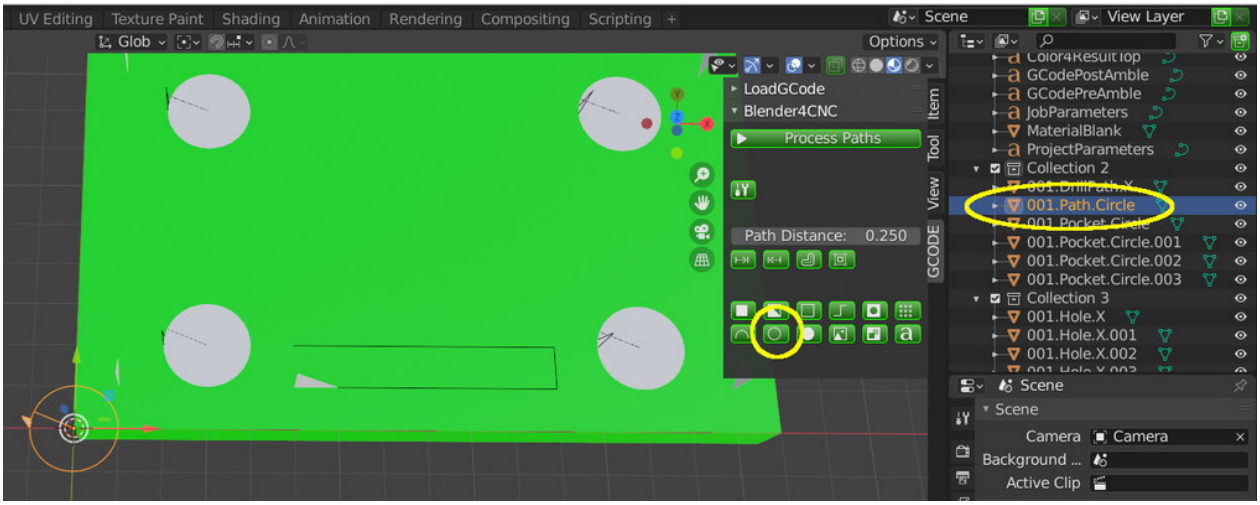

Figure 150: Let's get started creating the curved path. Click the "Create Circle Path" button.

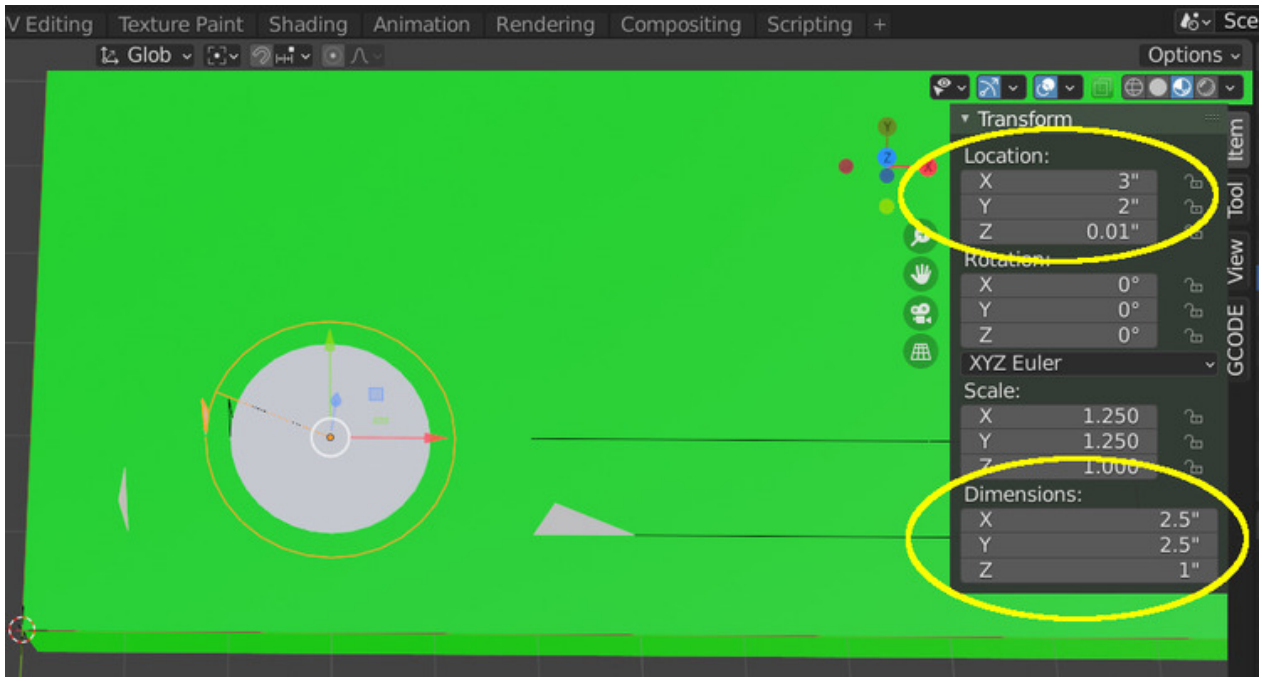

Figure 151: Move the circle path so it is centered on the circular pocket at (3,2) and resize it (dimensions X=2.5, Y=2.5).

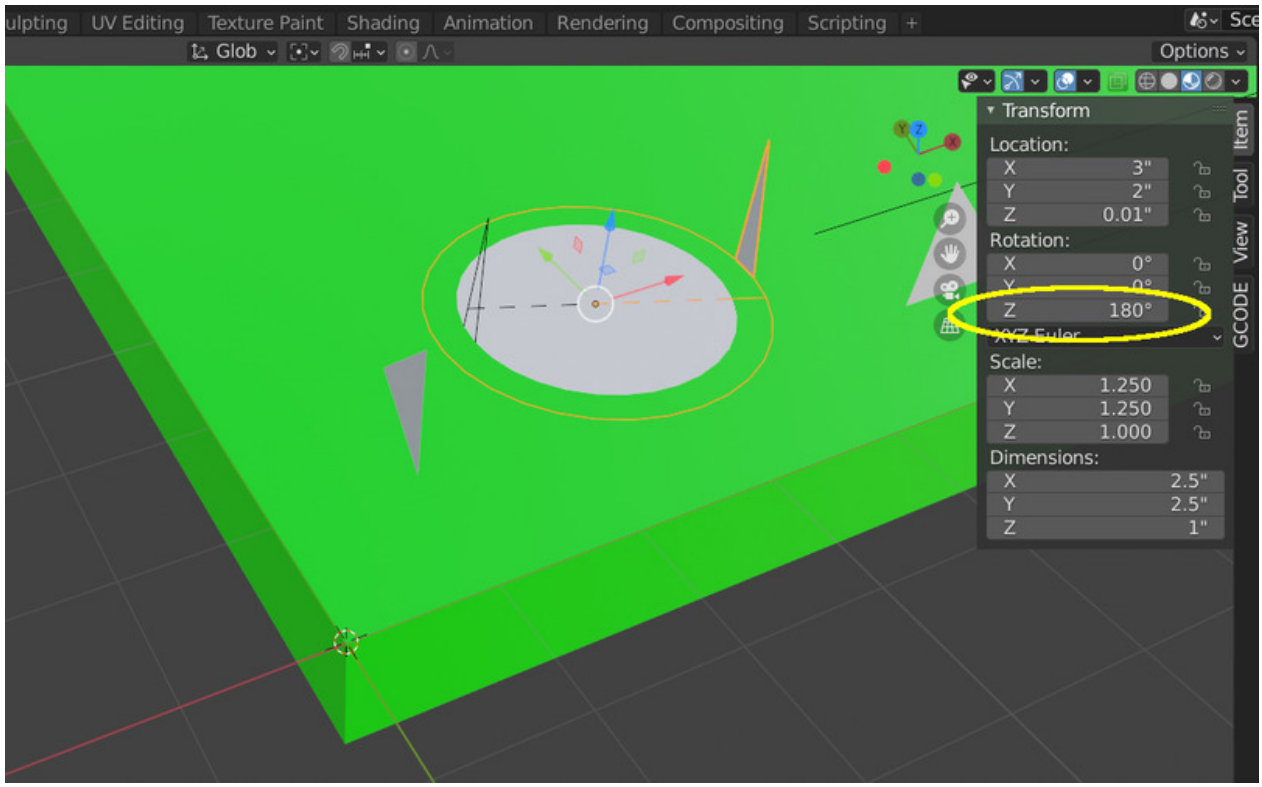

Figure 152: Rotate the operation 180 degrees (about the Z axis). By changing our perspective in the viewport, we can see that the "start" flag is now on the other side. We do this becuase we are about to delete points in the circle path and we want the start to be here.

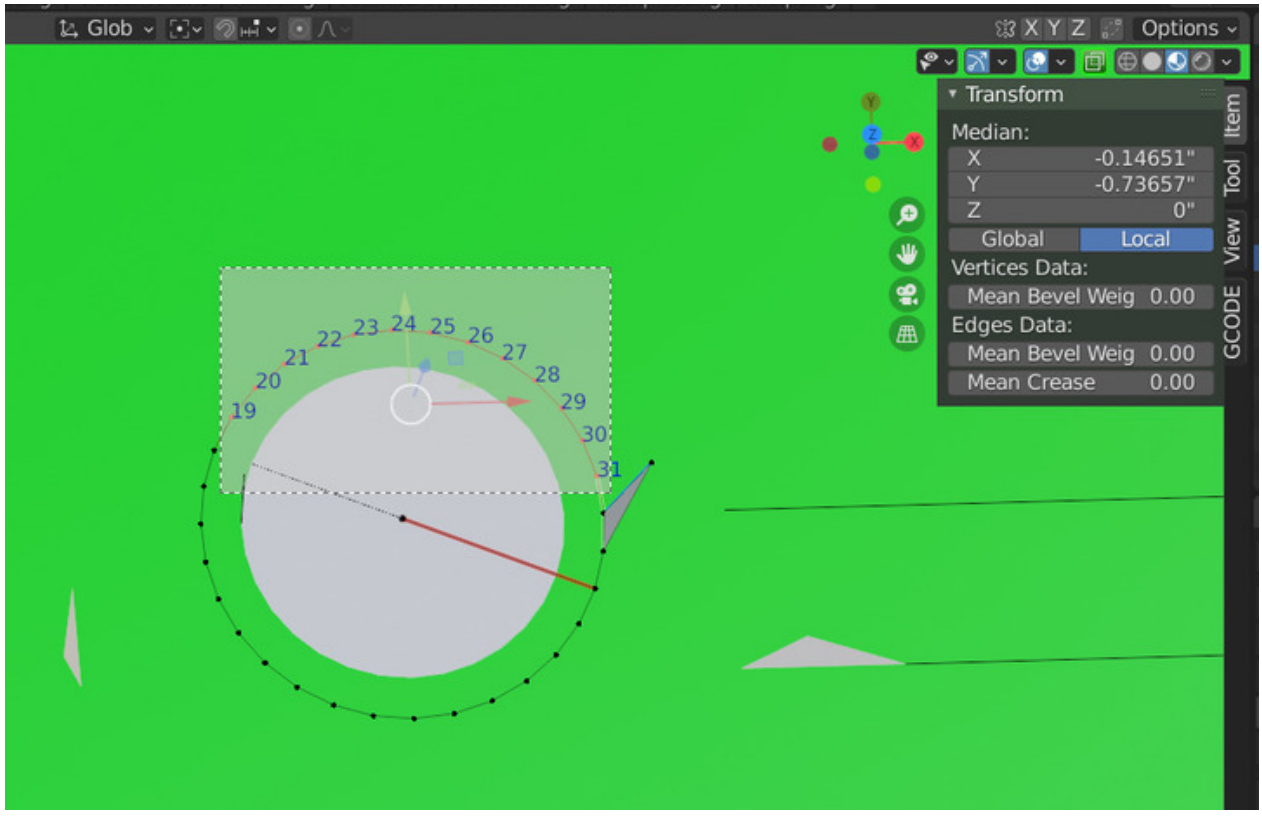

Figure 153: Press TAB to enter edit mode. Select all but 2 of the points in the "upper half" of the path.

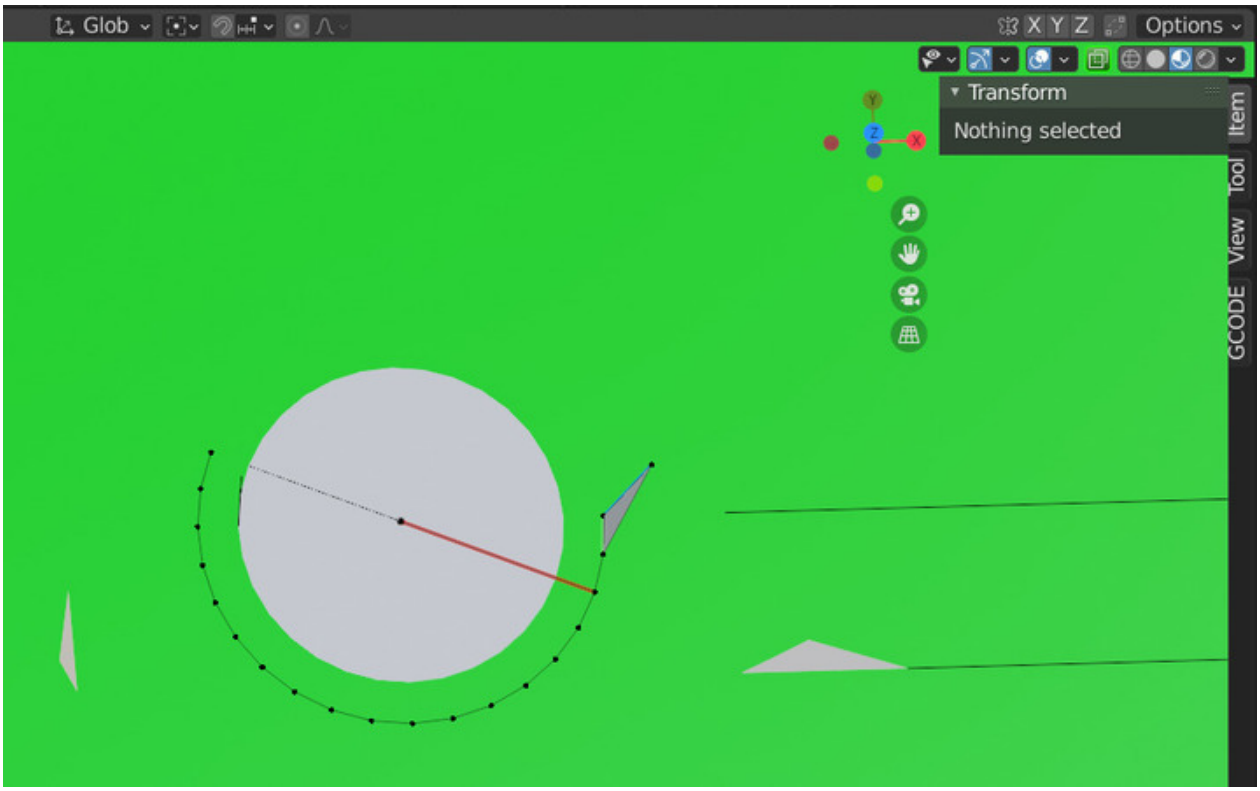

Figure 154: Press "Delete" and then select "Vertices" from the popup menu. And you should be left with a little over half the original circle.

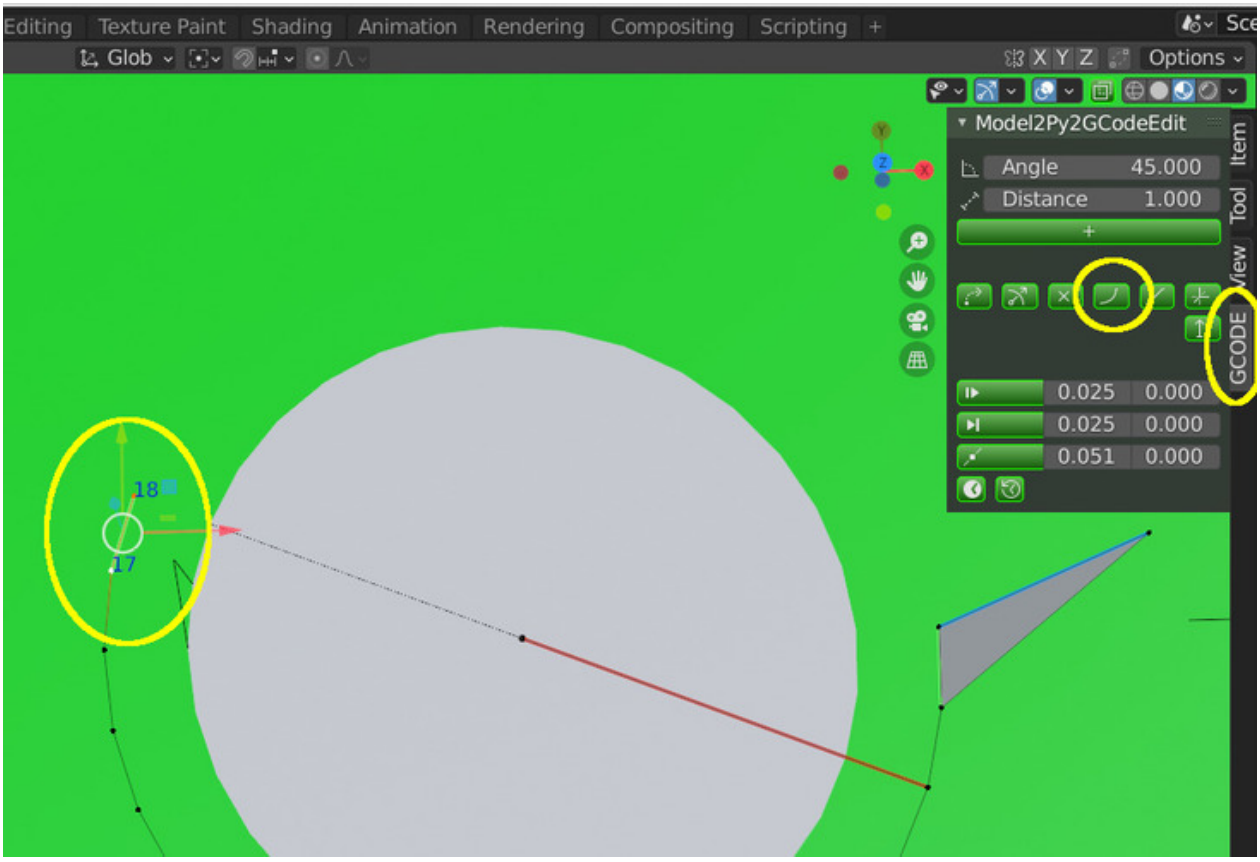

Figure 155: When we deleted the points from the circle, we lost the segment that marked the end of the curve. We must re-mark the end of the new curve. Select the last two points in the curve and then click the "Create Curve Segment" from the GCode tab. The last segment will turn a green color.

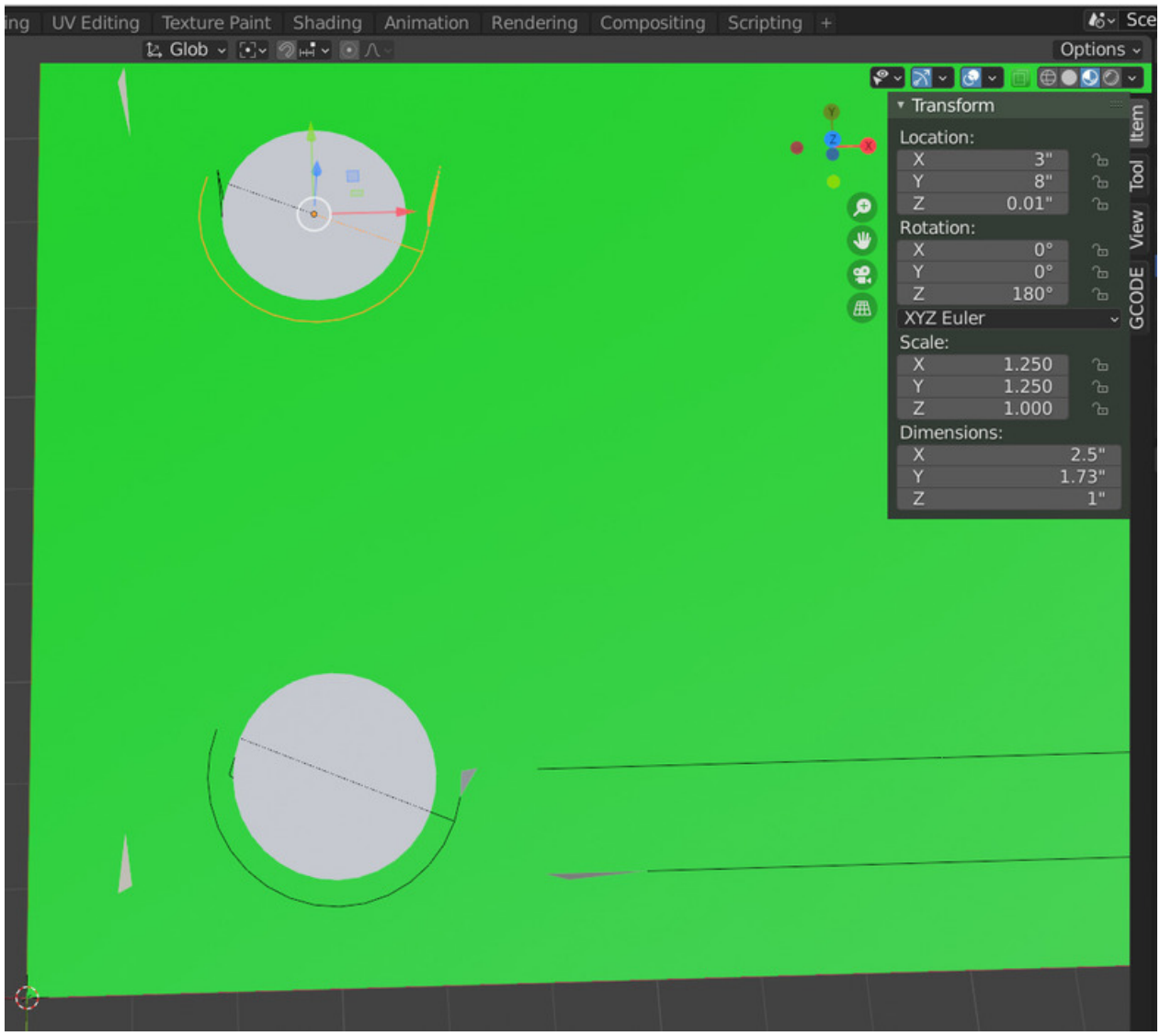

Figure 156: We are going to copy this whole curve to make the curve for the other end. Press TAB to exit edit mode. Now duplicate the curved path (press shift-d, then Enter). Move the new curve up to be centered on the circular pocket above at X,Y= (3,8).

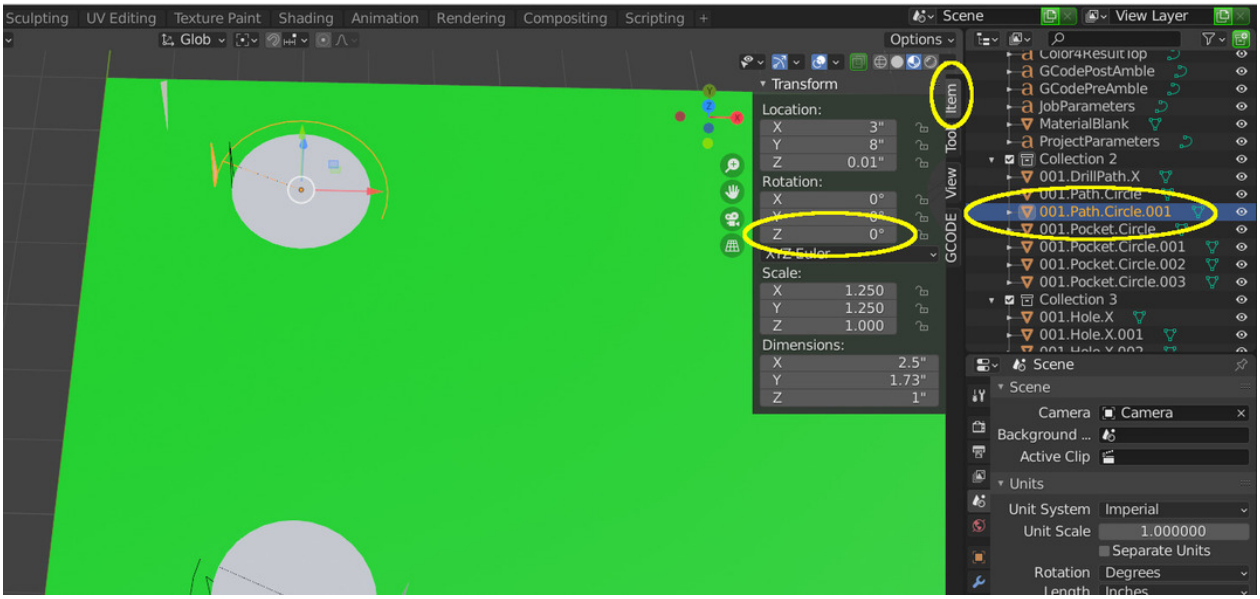

Figure 157: Rotate this operation back to 0 degrees (around the Z axis) by entering 0 into the Z rotation field. Note also on the upper right side of the display that we have two named paths because we are going to join these together in the next step into just one path.

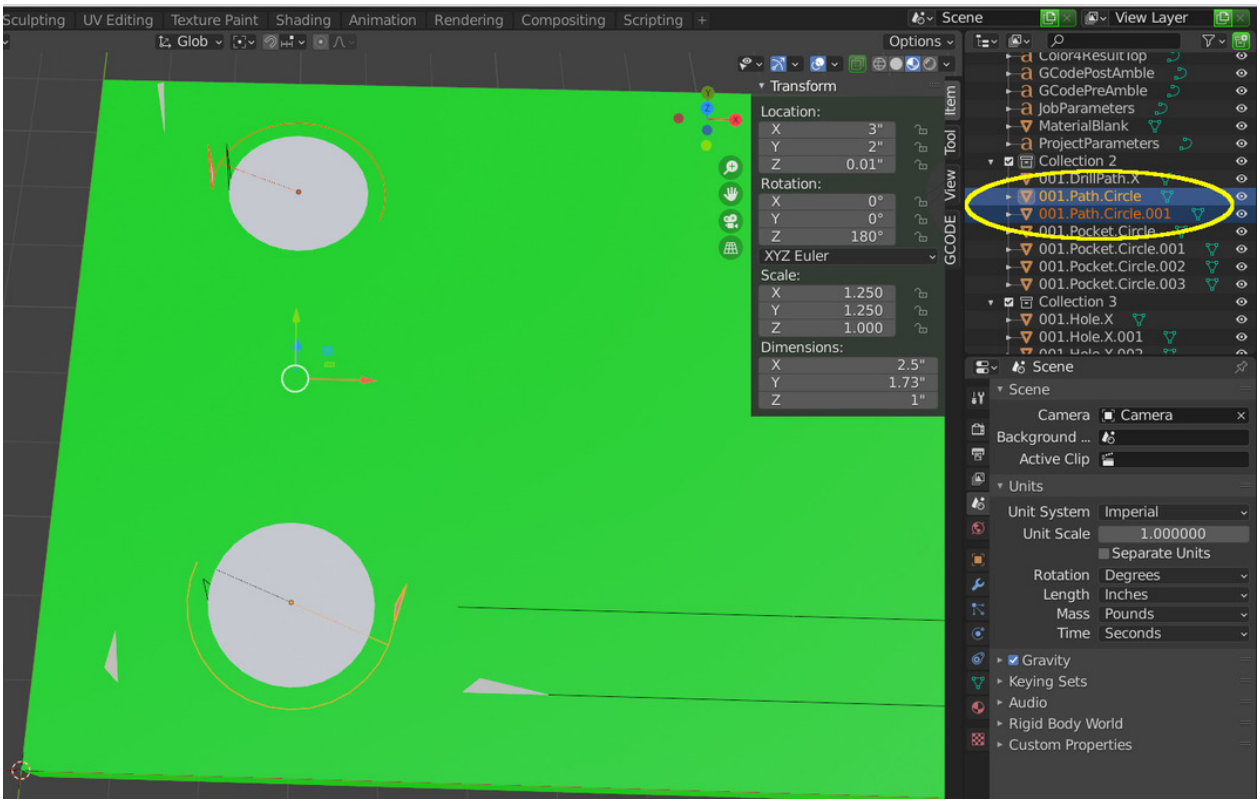

Figure 158: Select both curves (click the top one then shift-click the other lower curve). They should appear highlighted in orange in the viewport and in the list of names.

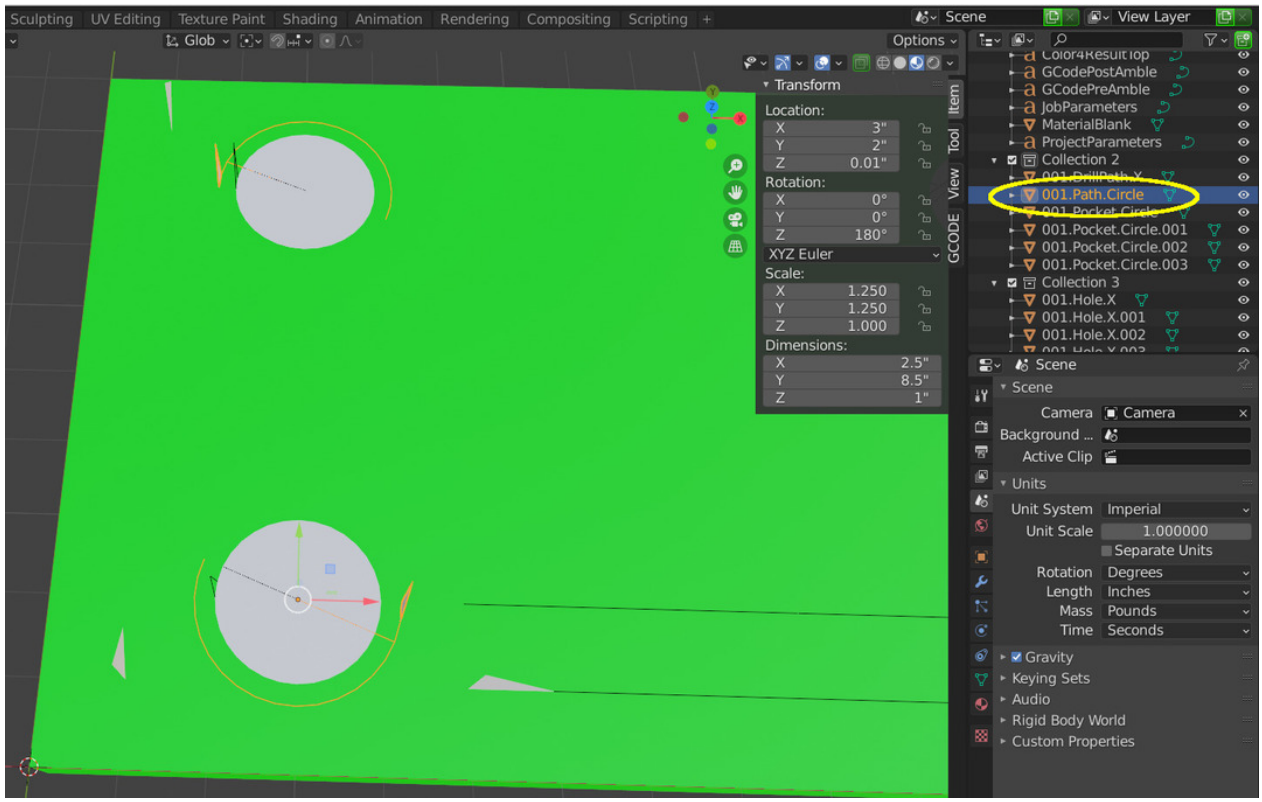

Figure 159: Join the curves together by pressing ctrl-j. You should only see one path operation now. (If you don't see the operations joined together, make sure your mouse is in the workspace 3D viewport window when you press ctrl-j.)

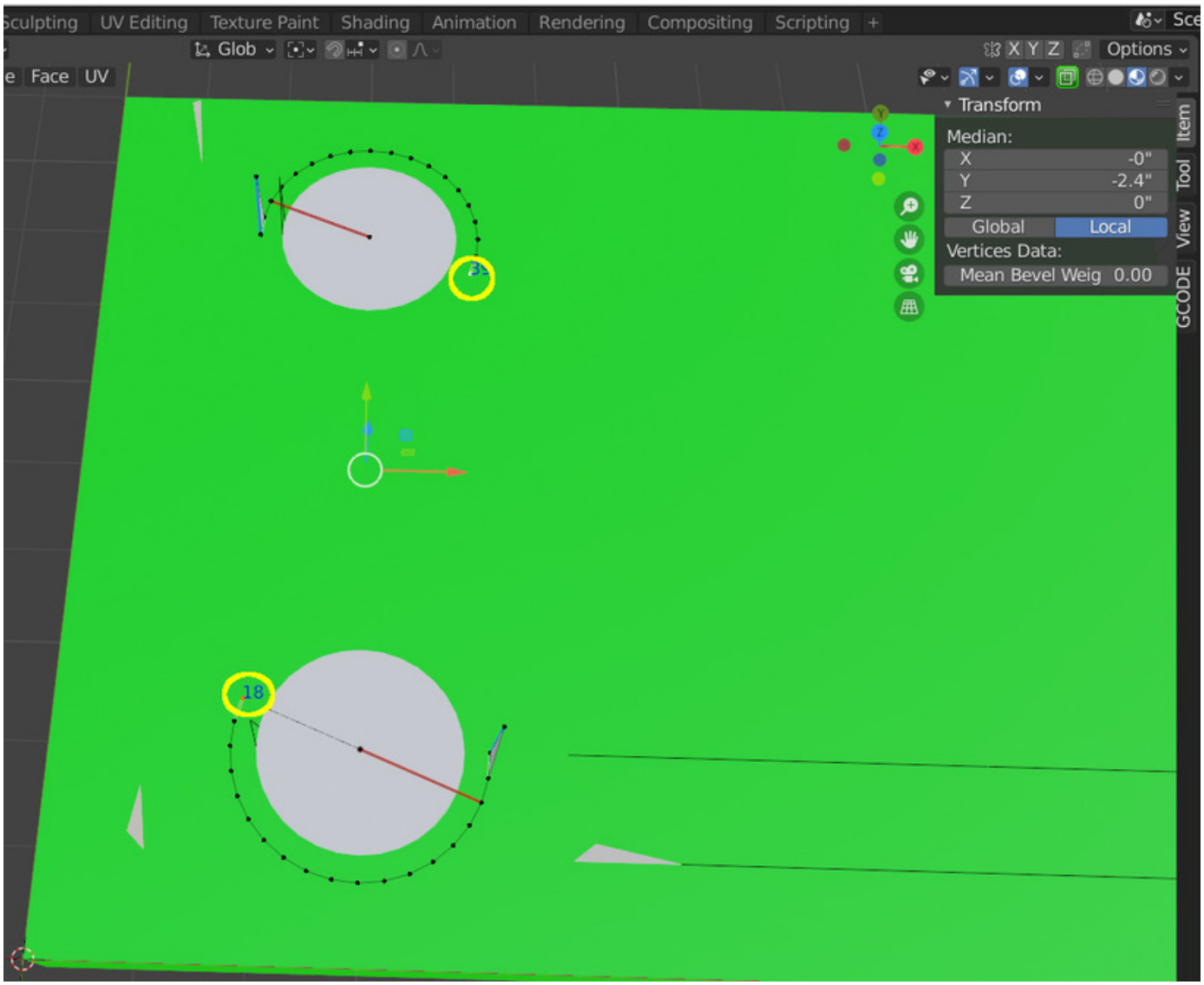

Figure 160: Press TAB to enter edit mode. The single path now includes both the original curves but it is disjoint and we must make it continuous. Select the two points shown and then press "f" to add an edge between them.

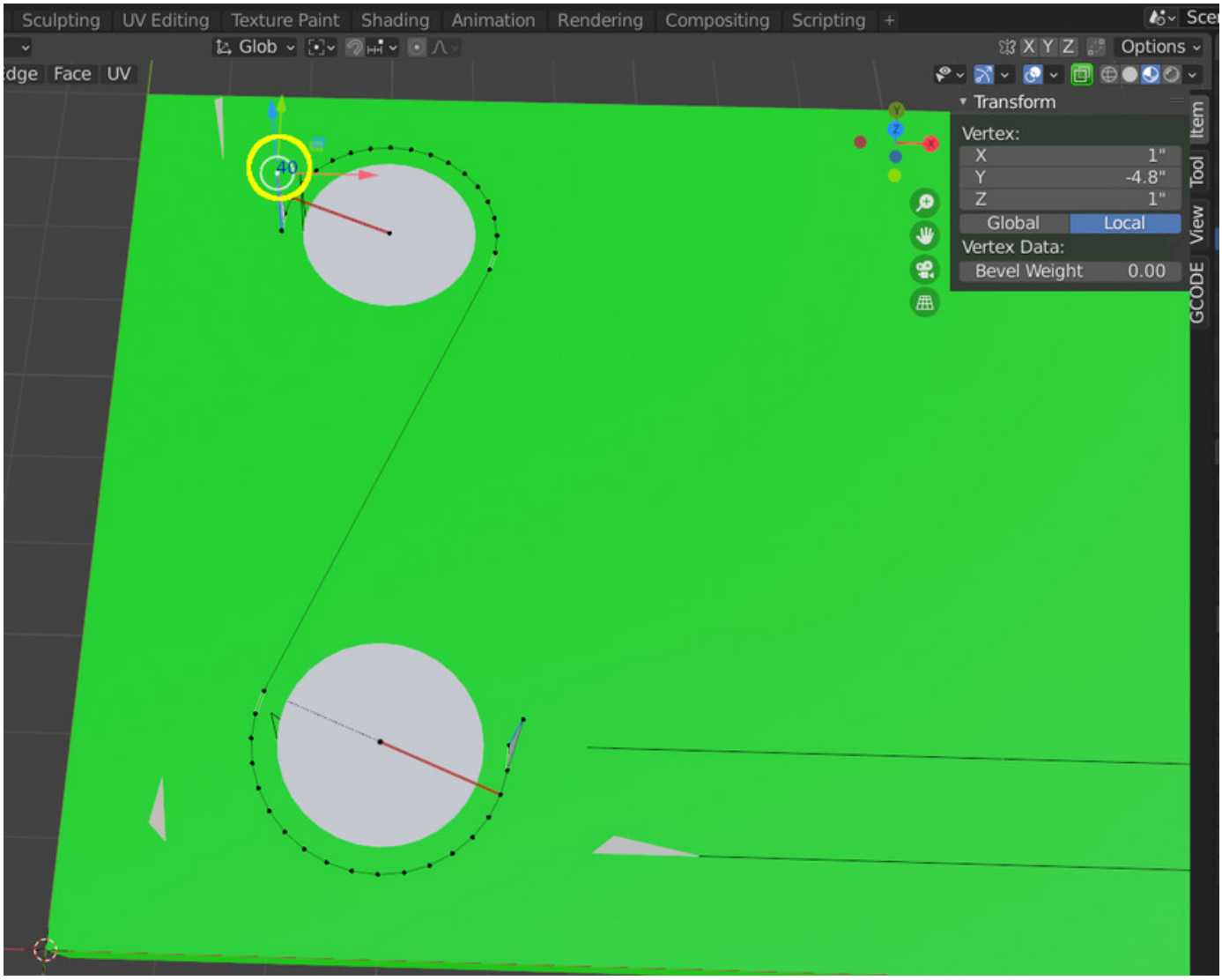

Figure 161: The path includes two start points and we only want one. So select the point at the top of the 2nd start point and press "Delete" then select "Vertices" from the popup menu.

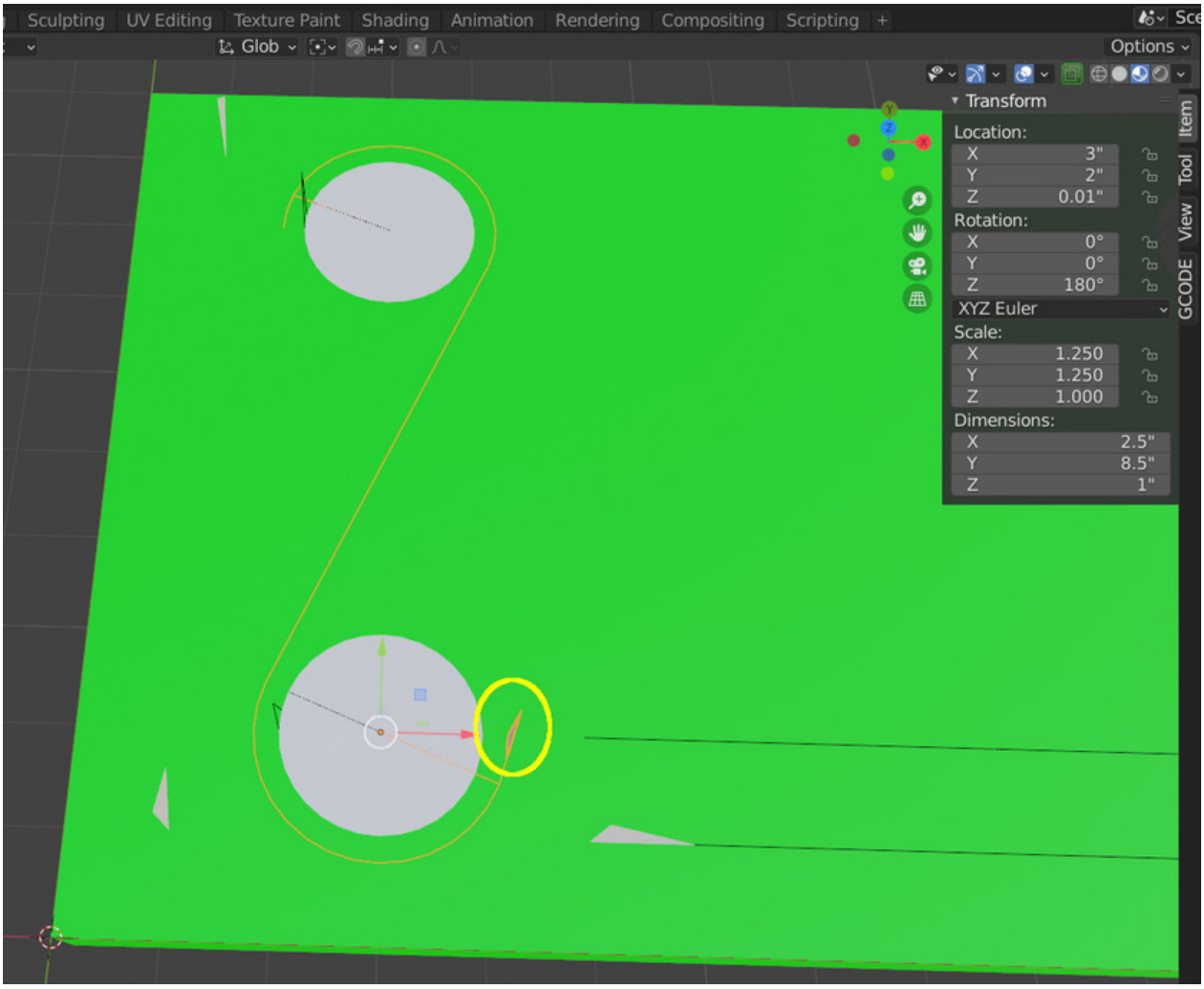

Figure 162: Press TAB to exit edit mode and you should see a nice continuous curved path that starts around the lower circular pocket, wrapping around the lower pocket and then going up and wrapping around the upper pocket.

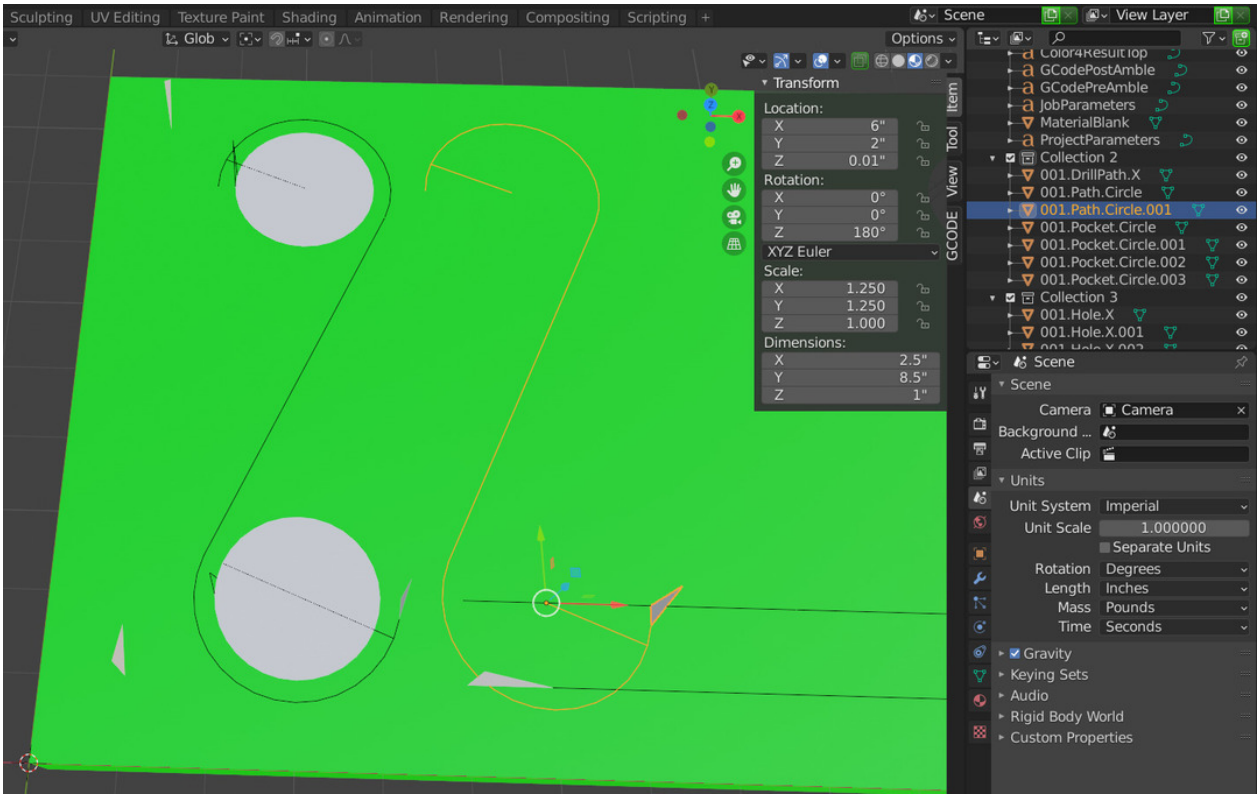

Figure 163: We now need to make a similar curve for the other side of the project but this will be easy because we can copy and flip the first one. Duplicate the curve (shift-d, then Enter) and move it over a little bit so we can see the new copy clearly.

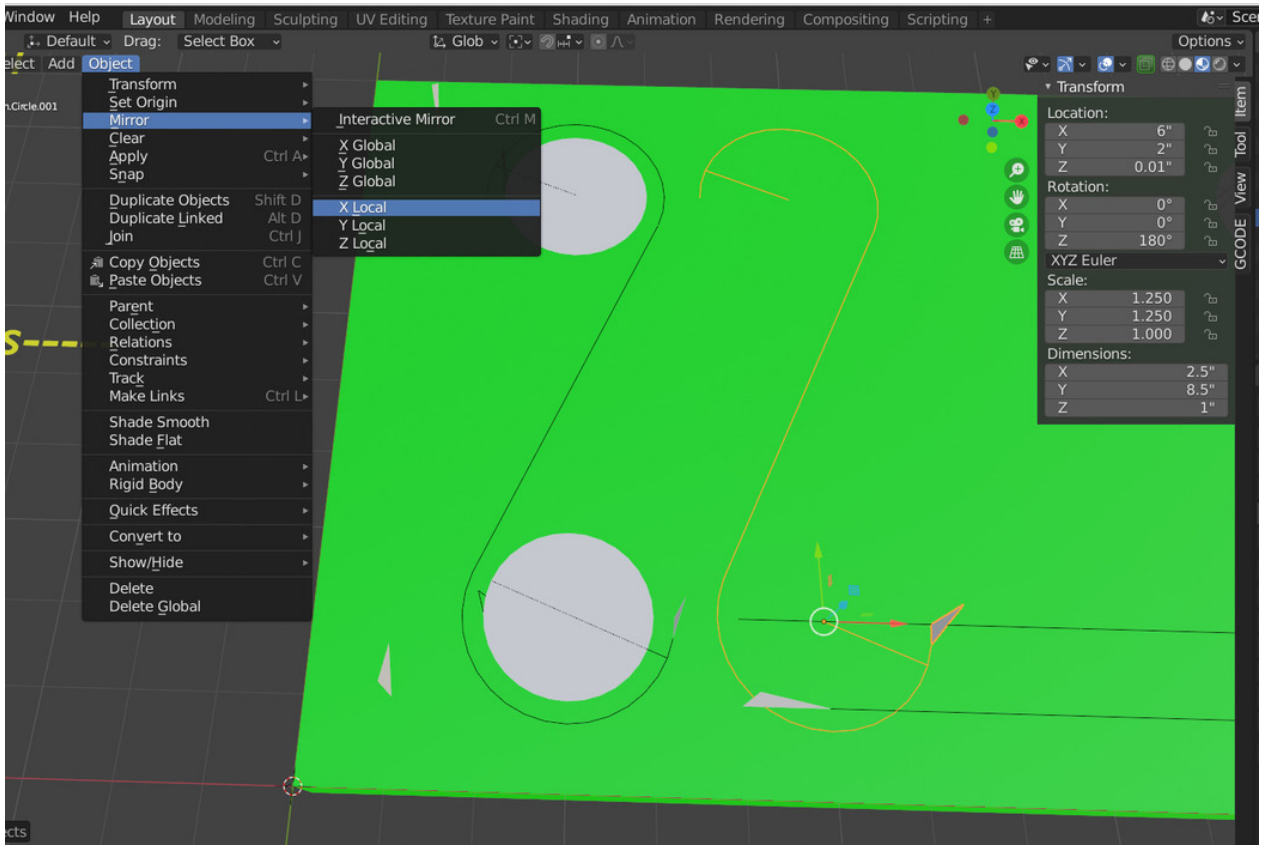

Figure 164: With the new copy selected, click on the "Object" menu, then "Mirror", then "X Local". This will flip the curve around like a mirror image.

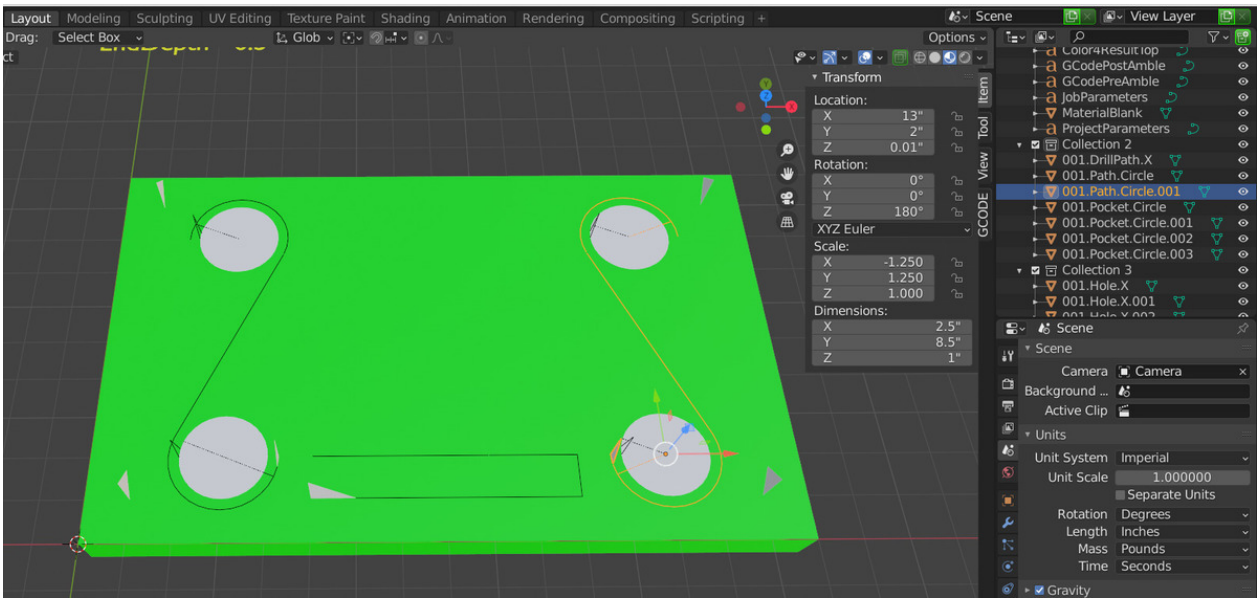

Figure 165: Move the mirrored curve over to wrap around the circular pockets on the right of the project.

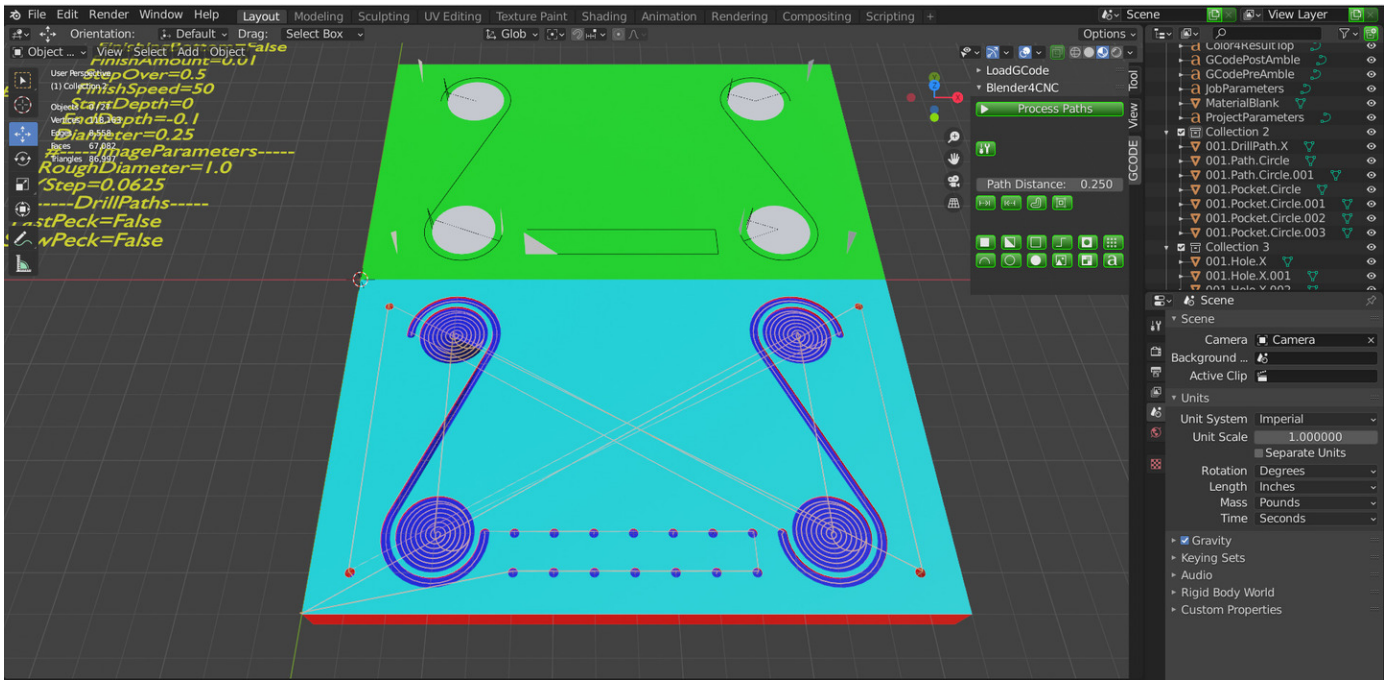

Figure 166: Click the "Process Paths" button to see how the project looks.

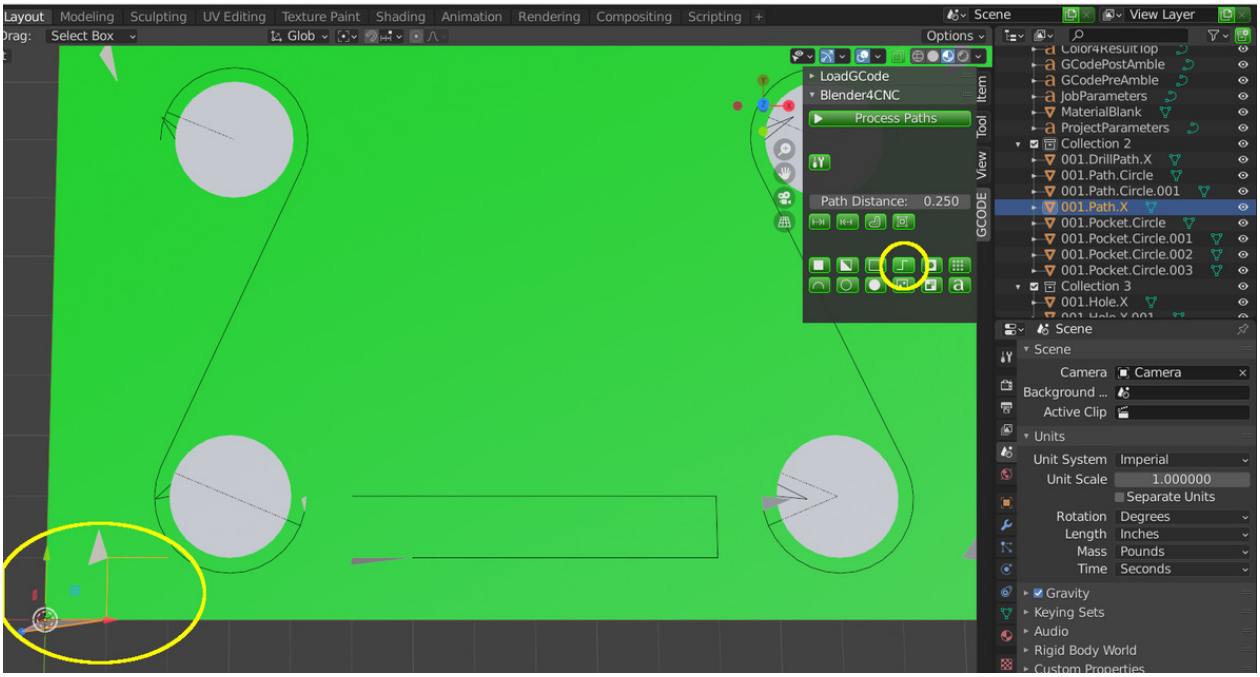

Figure 167: Let's add a couple of straight lines at the top of the project. Click the "Create Open Path" button.

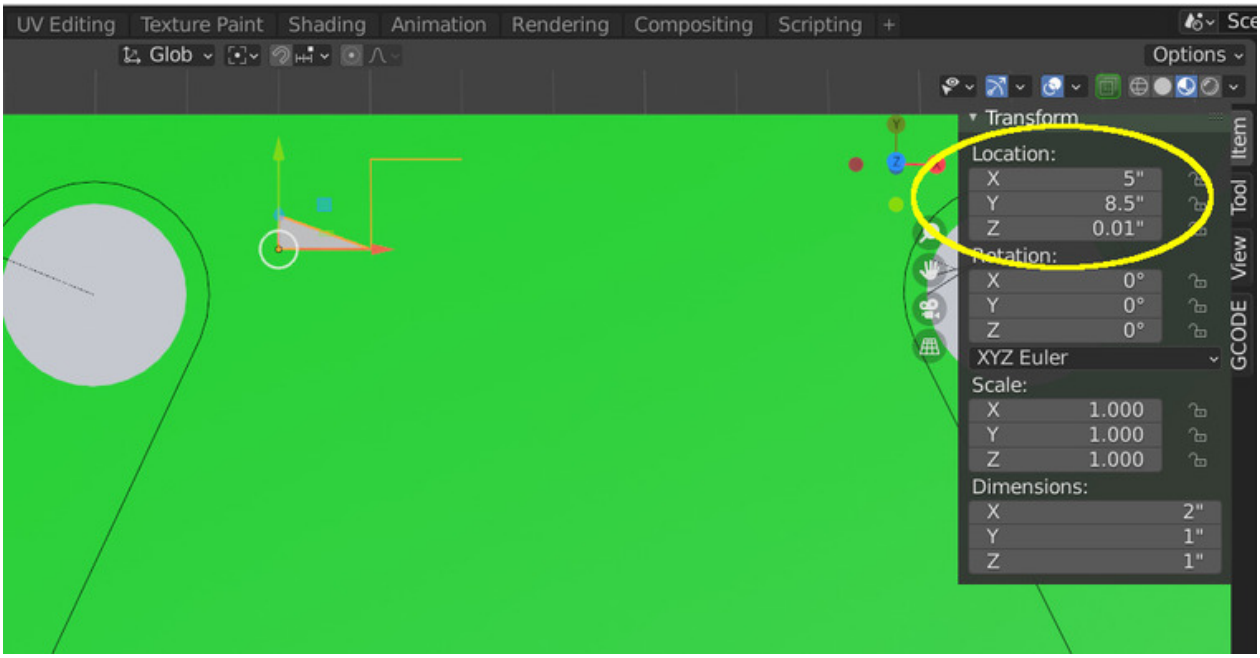

Figure 168: Move the new path object.

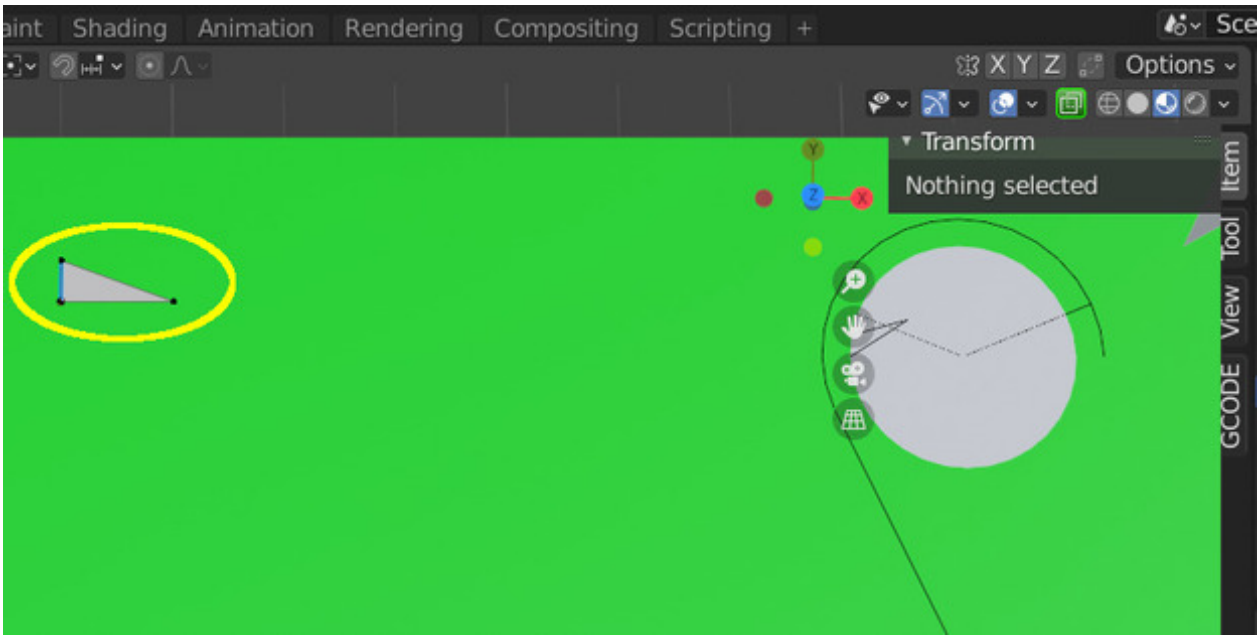

Figure 169: Delete the top two points (press Tab to edit, select top 2 vertices, press DEL and select "Vertices" from the pop-up) so we are left with just two points (and the 3rd point at the "top" of the start).

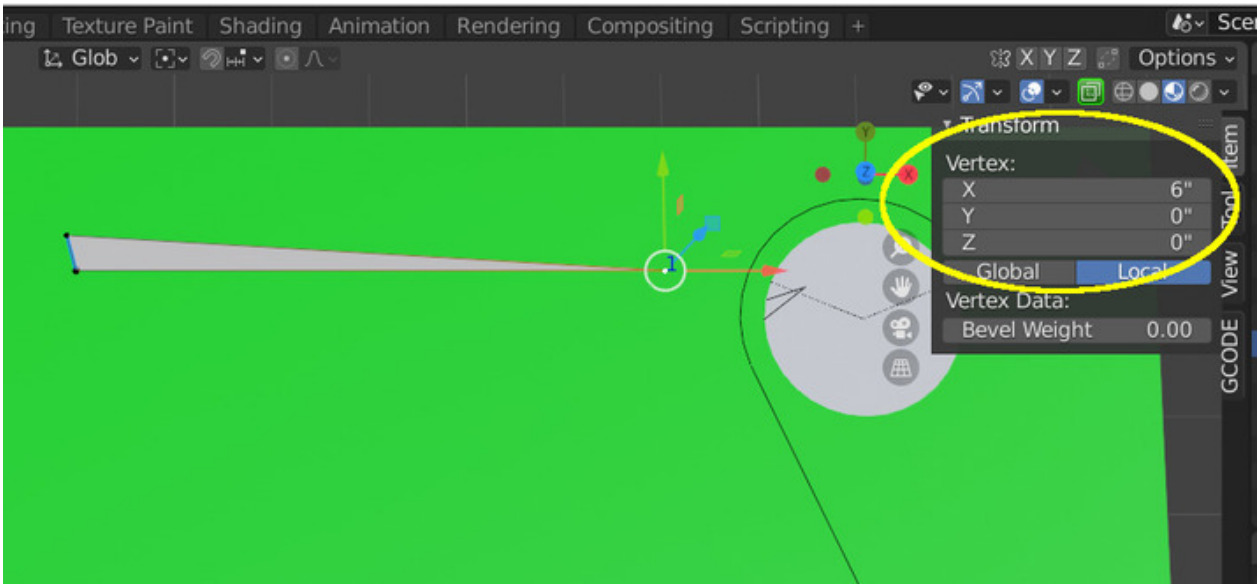

Figure 170: Select the "2nd" point and move it to (6,0).

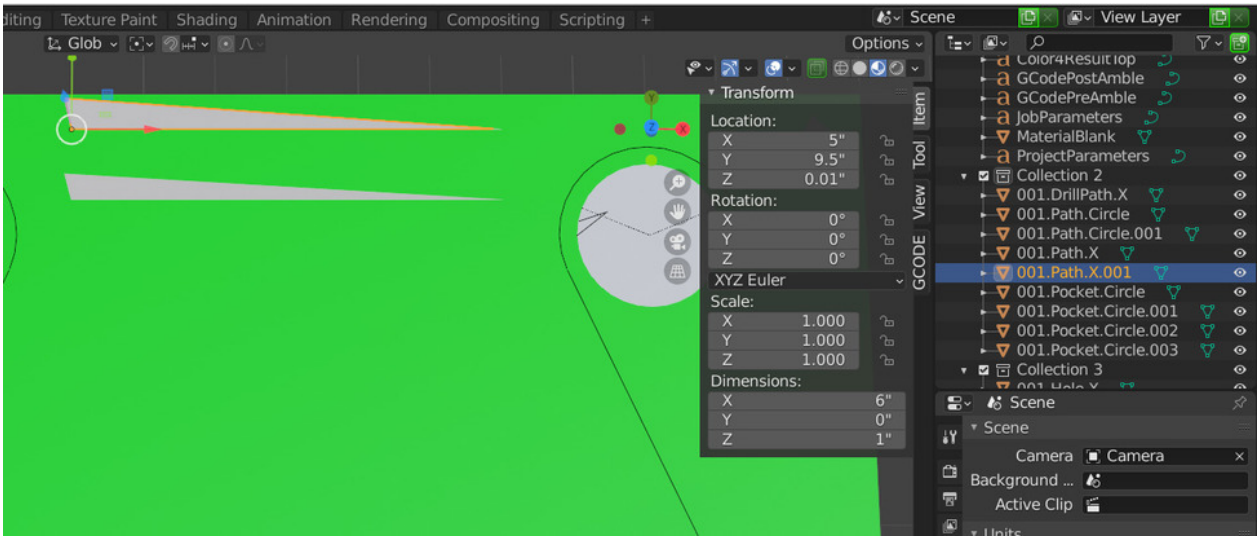

Figure 171: Press TAB to exit edit mode then duplicate the path (shift-d, then Enter). Then drag the copy up to (5,9.5) so we have two separate lines of the same length.

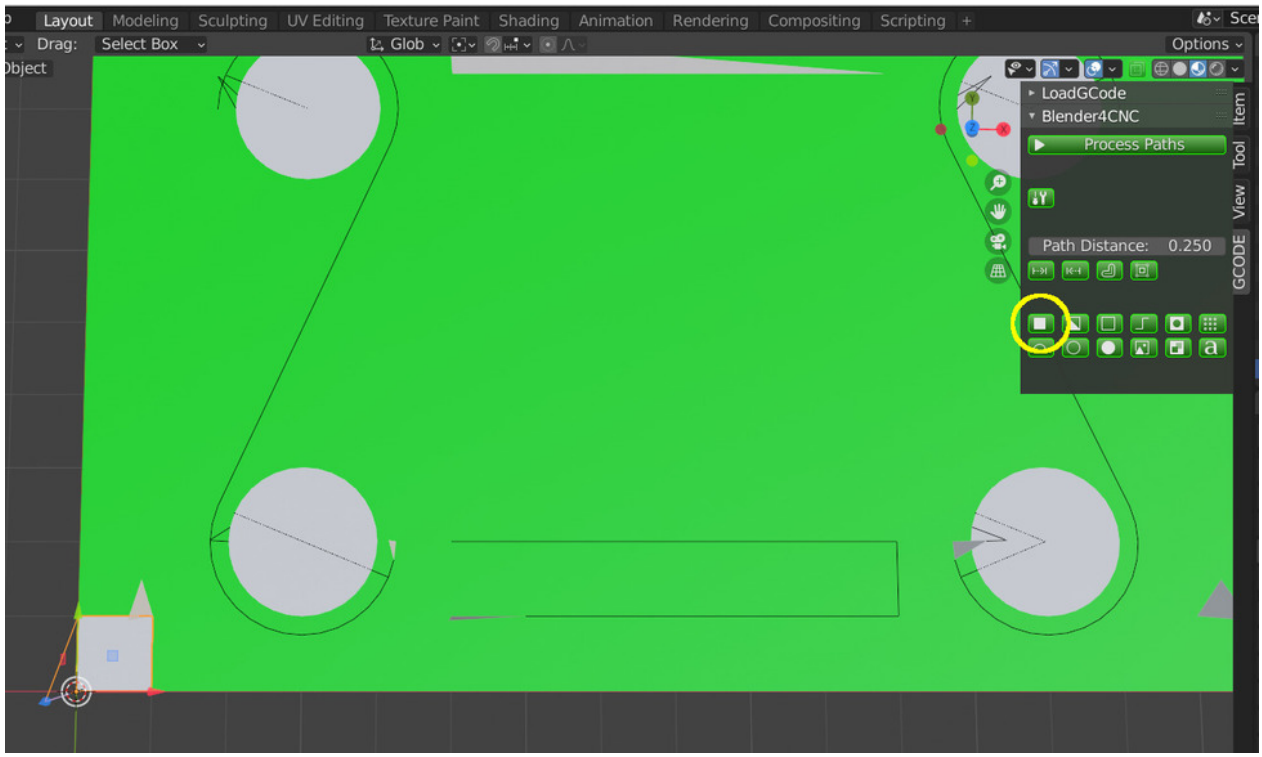

Figure 172: Click the "Create CW Pocket" button to add a pocket.

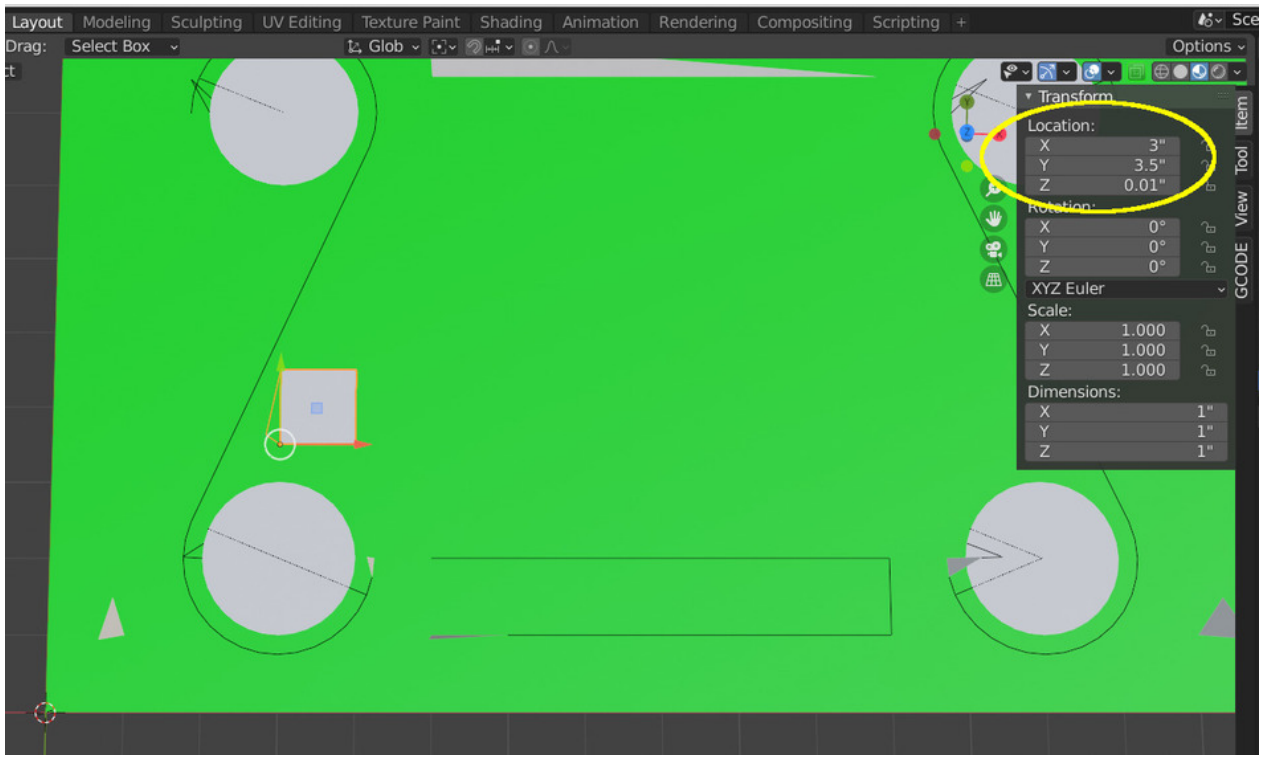

Figure 173: Move the pocket to the shown location.

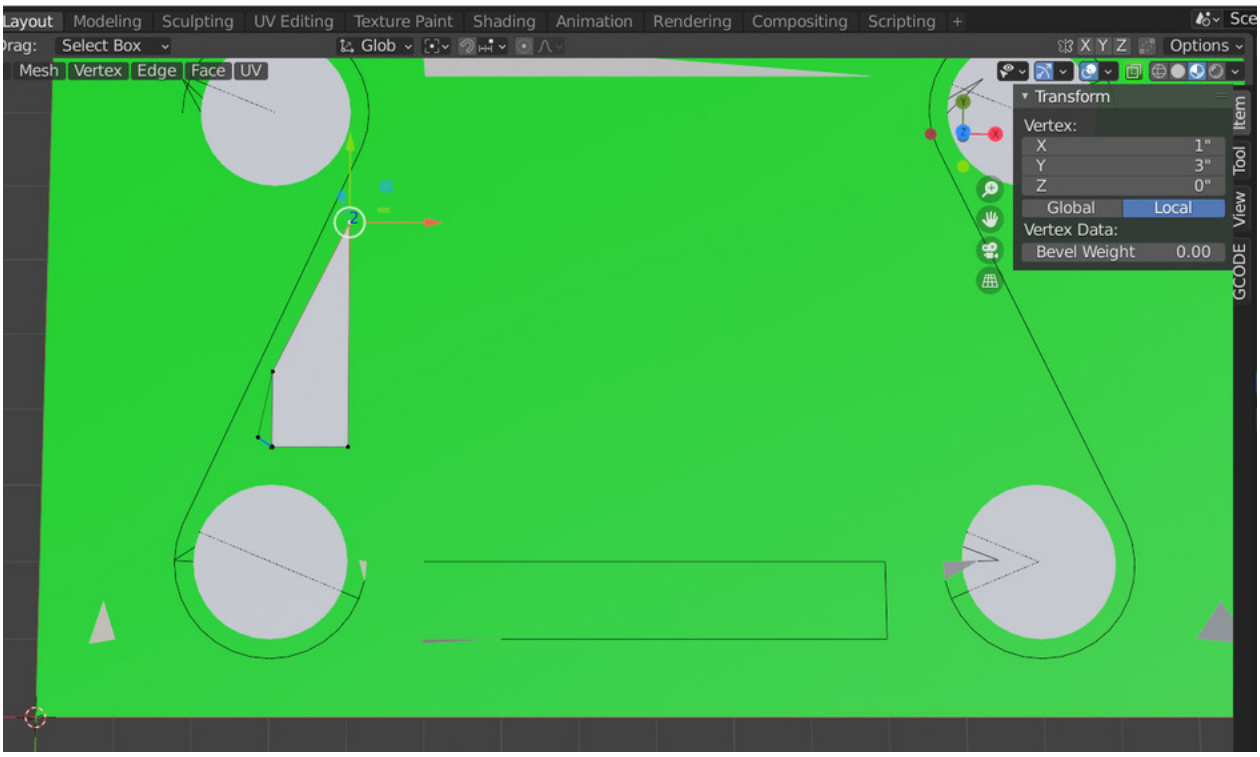

Figure 174: Enter edit mode (TAB), click off of the object to deselect all vertices, then click the top right vertice, and drag the corner point up.

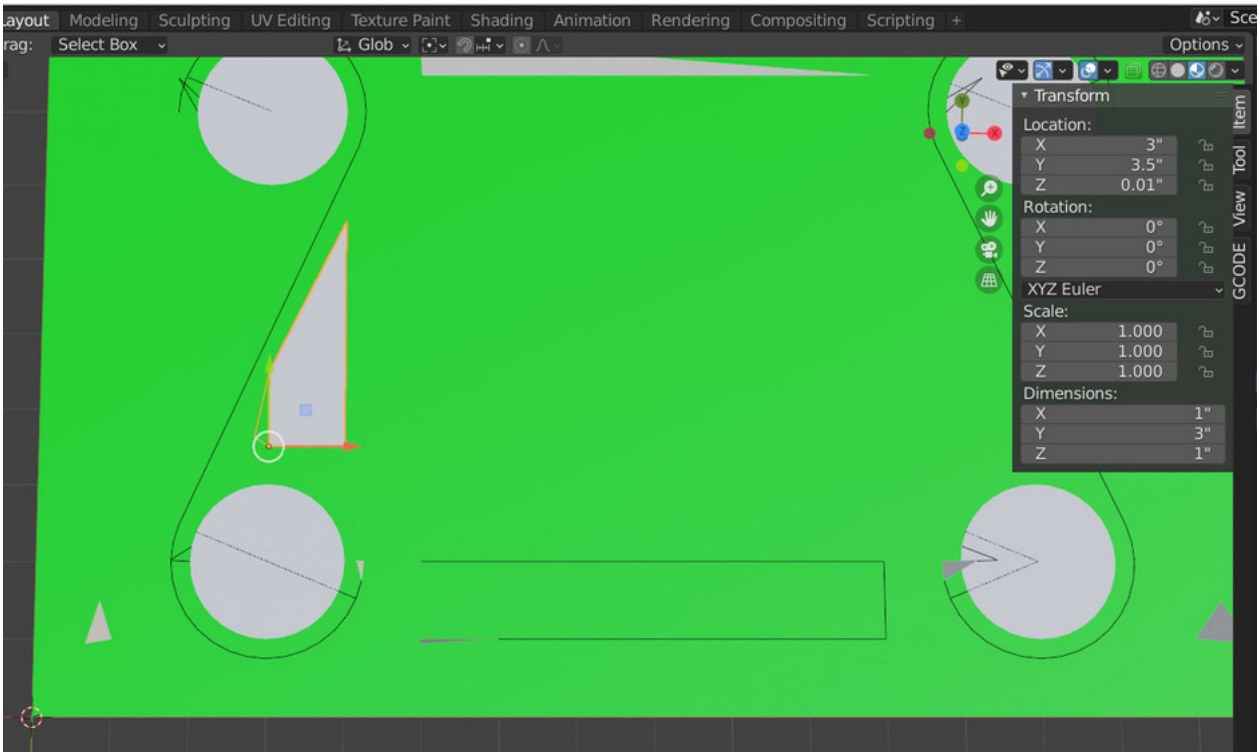

Figure 175: Exit edit mode (TAB).

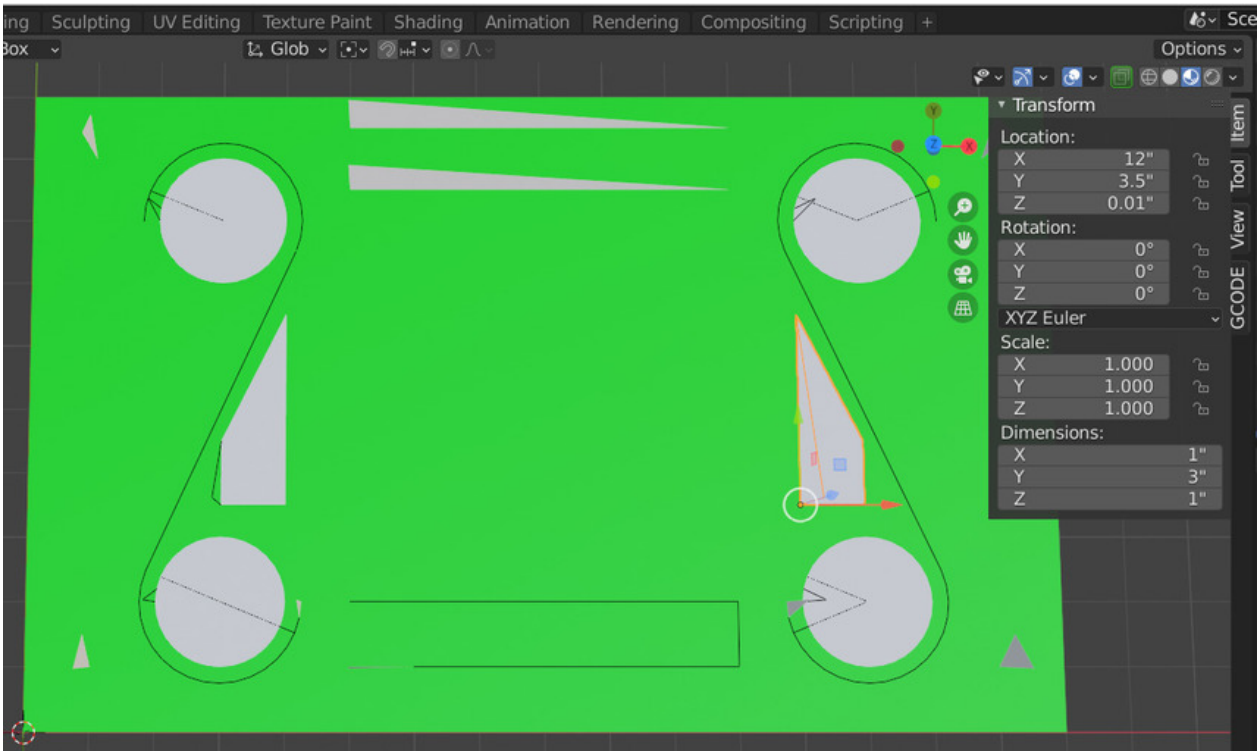

Figure 176: Repeat the previous few steps to produce another pocket on the right hand side of the project.

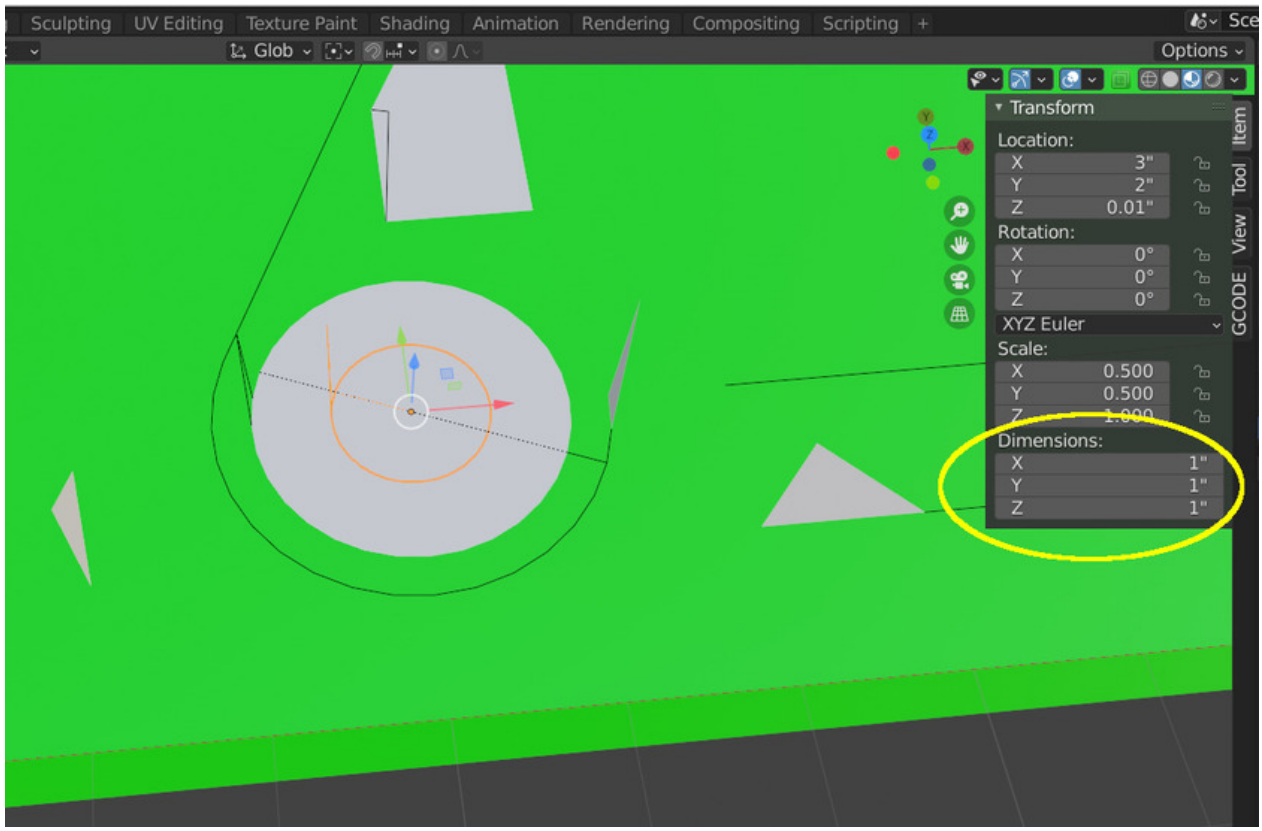

Figure 177: Let's create a couple of deep circular pockets within the two lower circular pockets. Start by selecting the circular pocket in the lower left and duplicating it (shift-d, then Enter). Change the dimensions to 1.0x1.0. Now we can see have a smaller "circle" within the original circular pocket we copied.

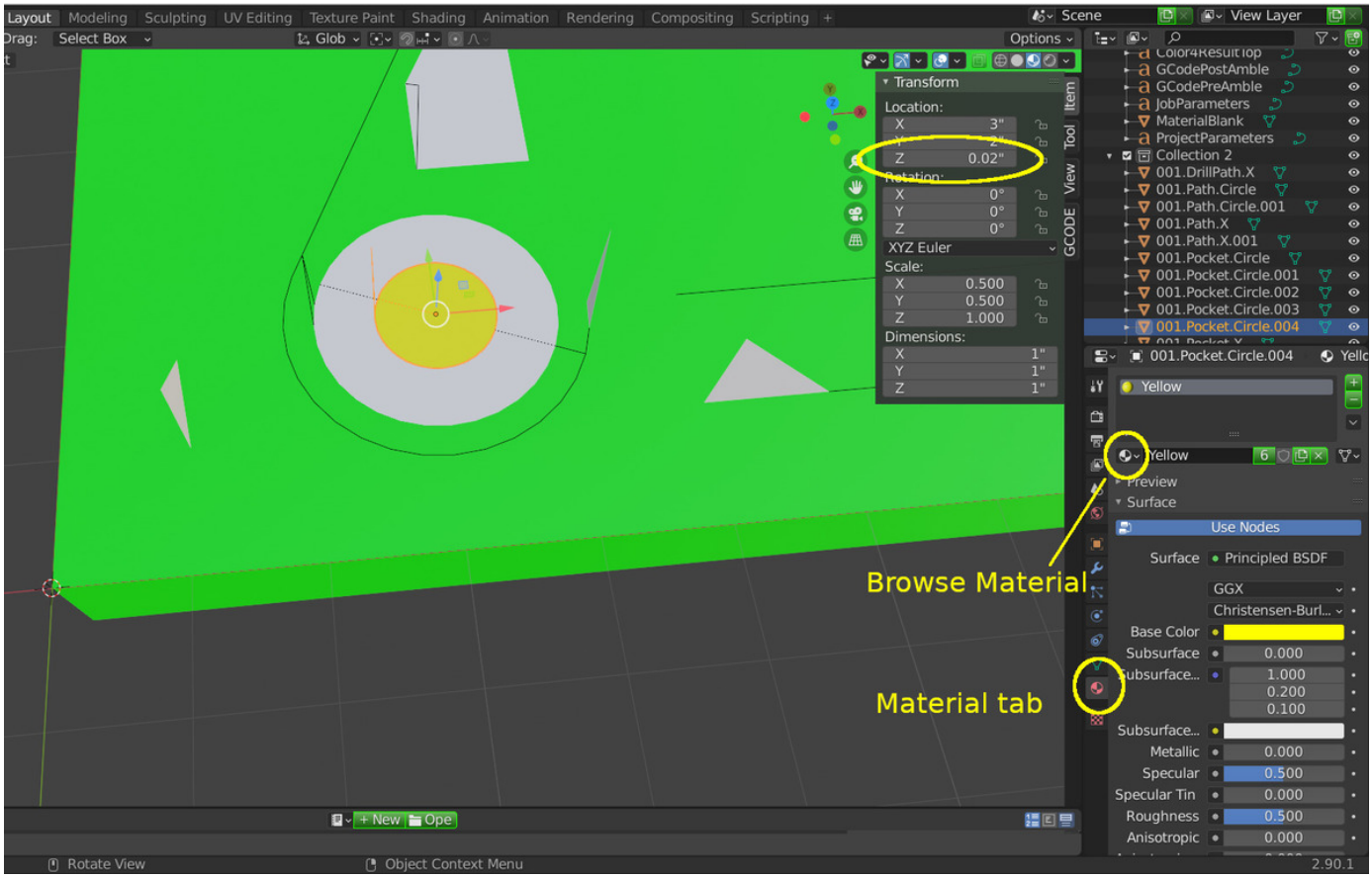

Figure 178: Because the new pocket blends with the original pocket (gray on gray) let's change the material just so that we can see it stand out. Click on the "Material" tab in the lower right of the display. Then click the "Browse Material to be linked" button and select a material. In this case I will choose yellow. (You could also click on the "+New" button and create a new material.) Also, set the Z location to 0.02".

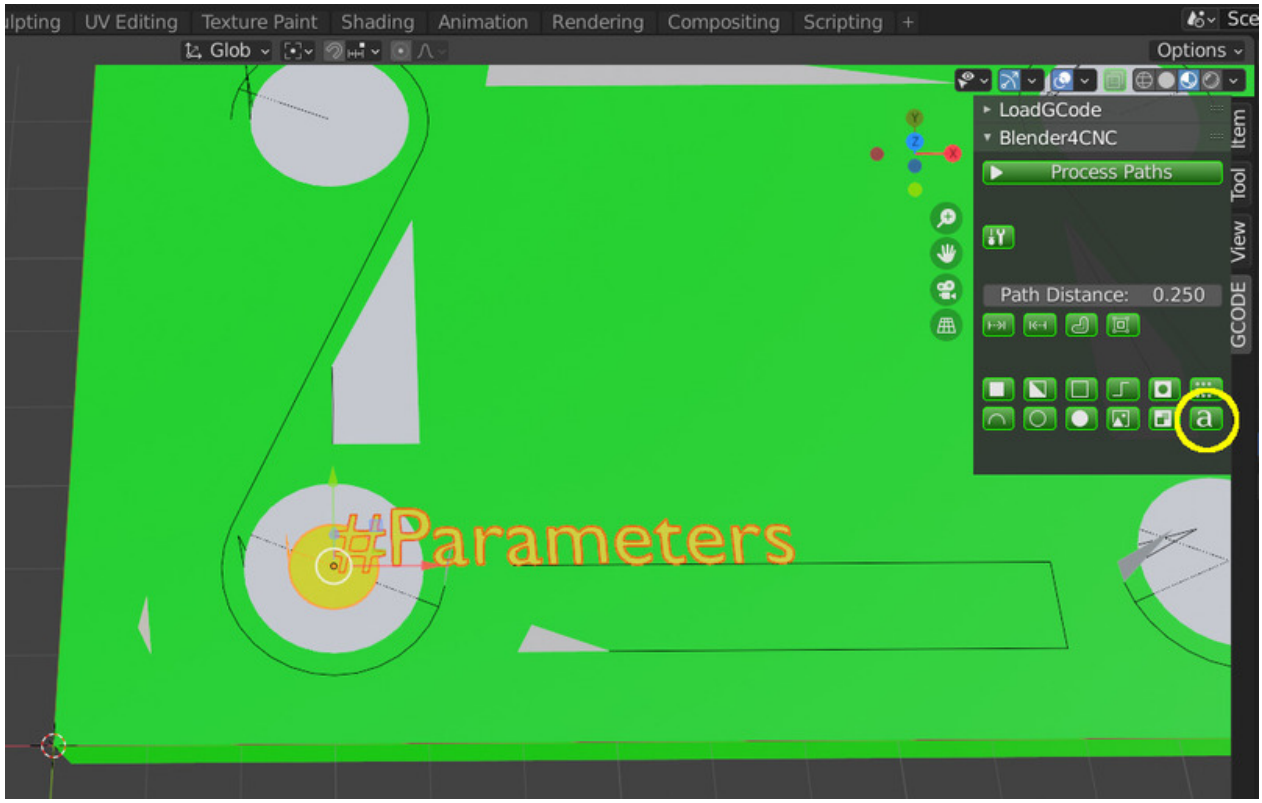

Figure 179: To make this pocket cut deeper, we are going to have to override the default end depth of -0.1. Click the "Create Parameters" button and you will see a text object appear that contains the text "#Parameters". This new text object is linked as a "child" to the pocket operation.

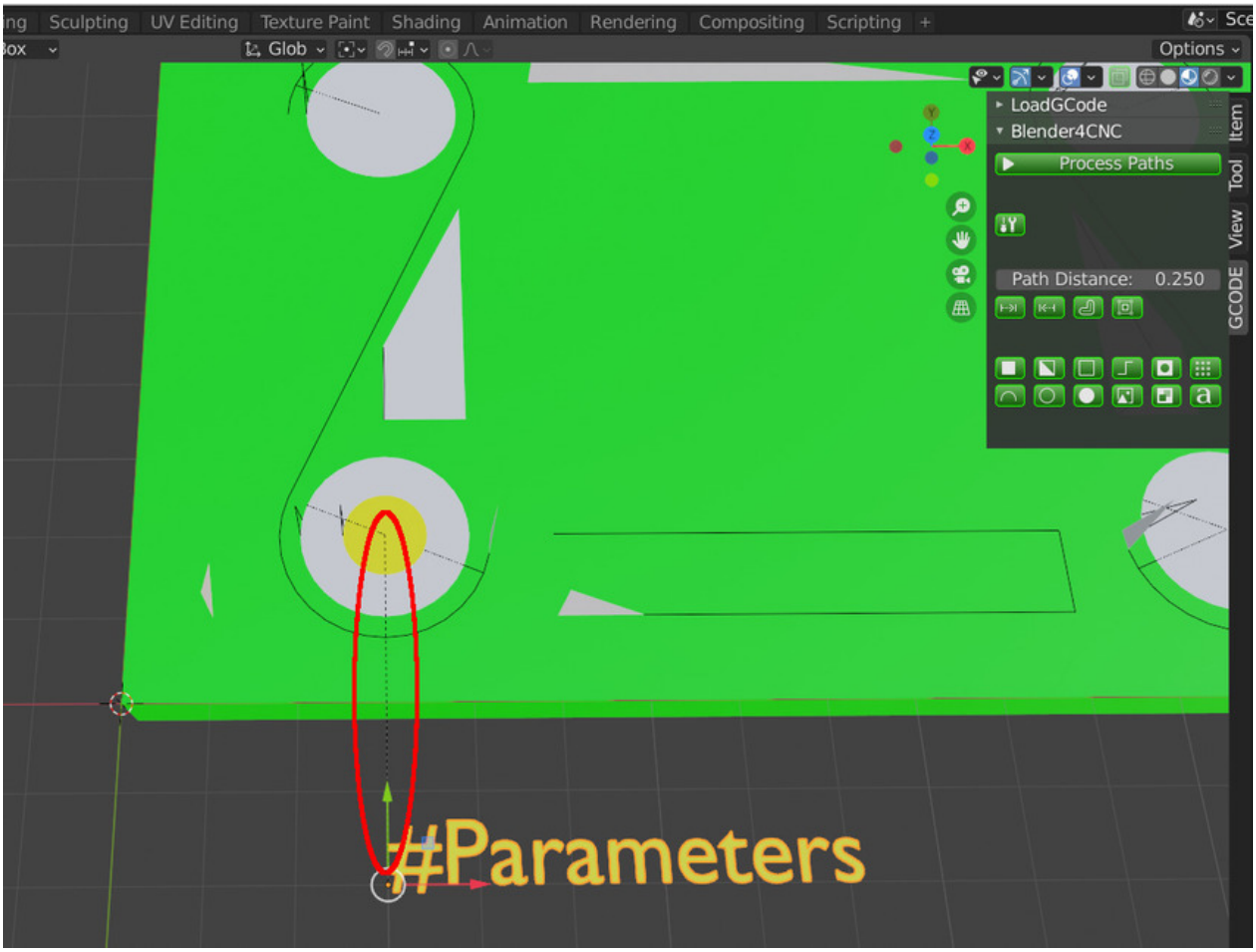

Figure 180: Move the parameters down to see it clearly and you will notice that there is a dotted line connecting it to the pocket operation (this is how you know it is linked as a child object). It is difficult to see the dotted line though against the background and the colors of other objects.

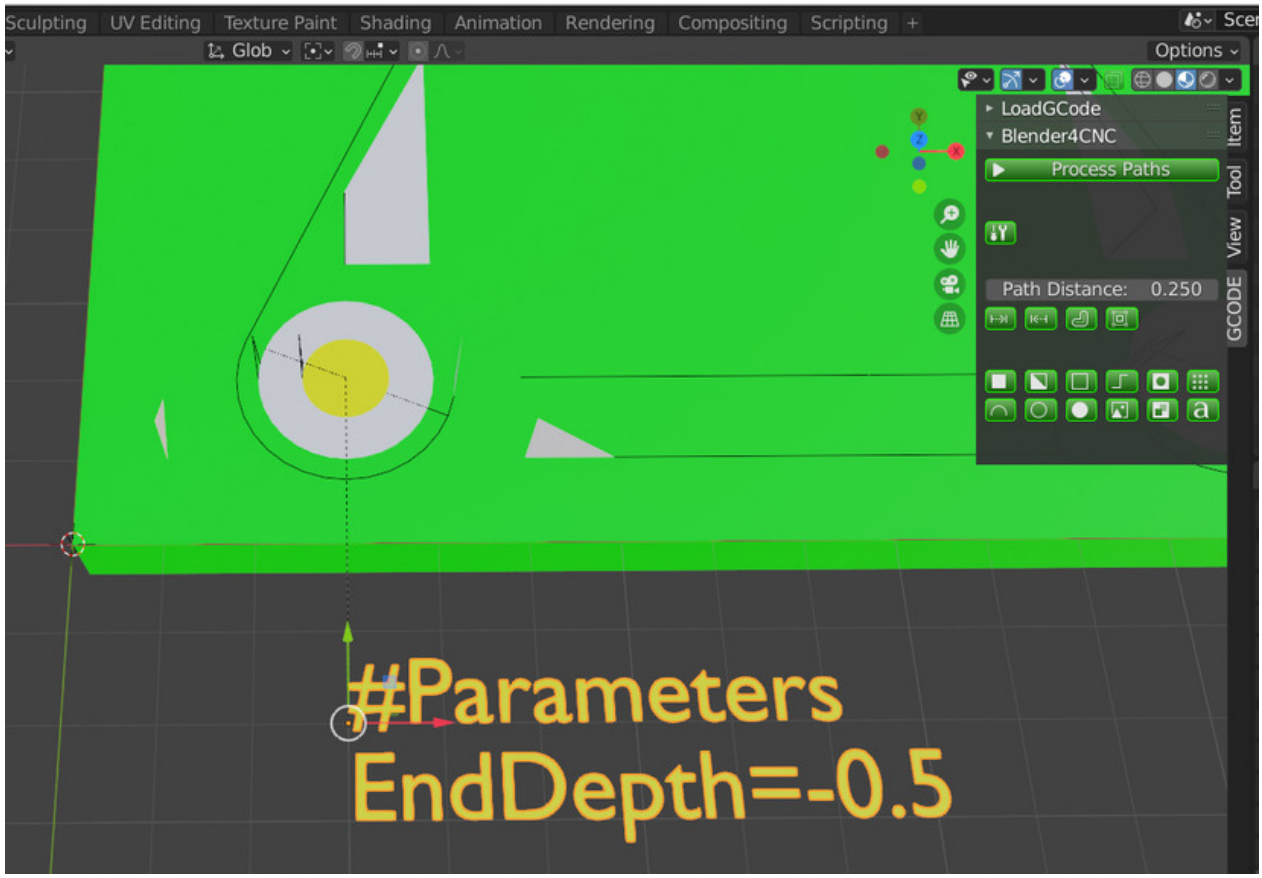

Figure 181: Press TAB to edit the text and add the text "EndDepth=-0.5" as shown. Then press TAB again to exit edit mode.

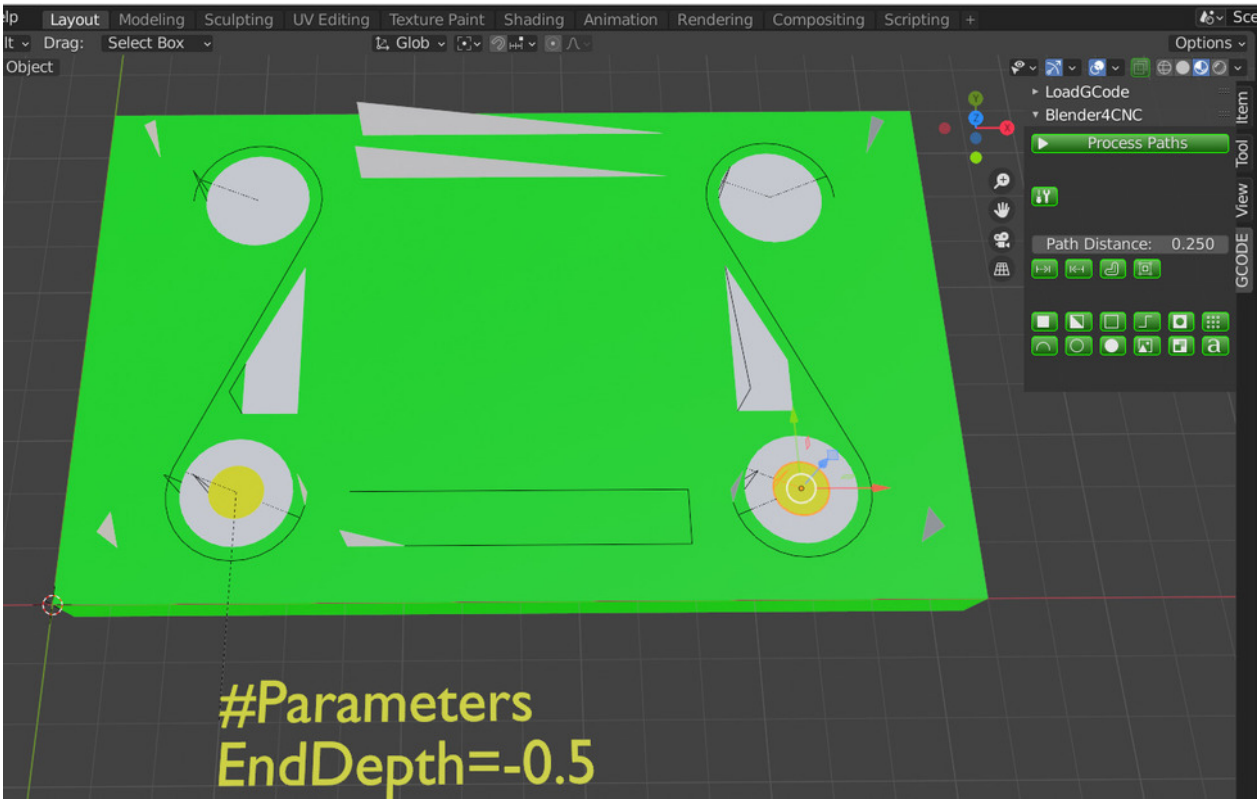

Figure 182: Let's create another deep circular pocket on the right side of the project. Select the deep (yellow) pocket and duplicate it (shift-d, then Enter). Move it over to be in the center of the pocket on the right.

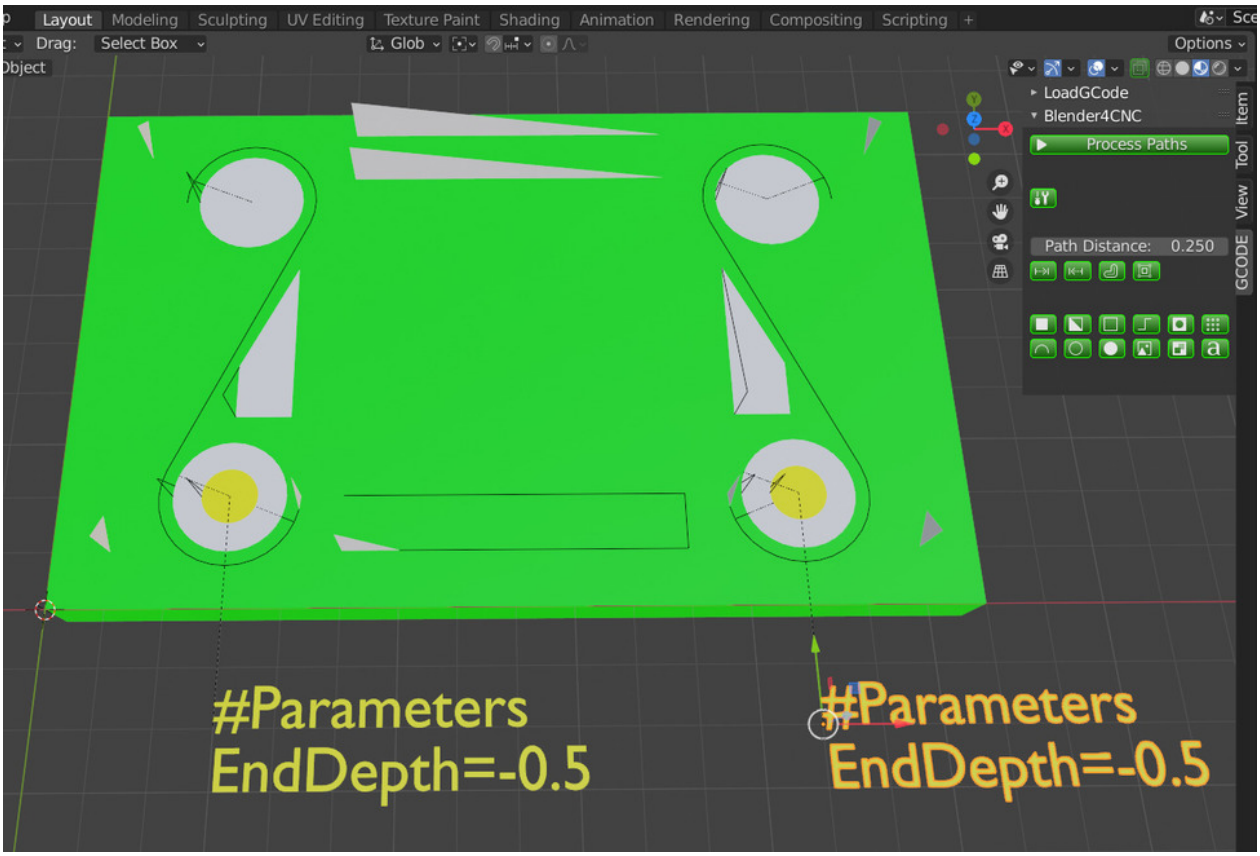

Figure 183: Repeat the previous steps to create a parameters object to override the depth. (Click the "Create Parameters" button and then edit the parameters text.)

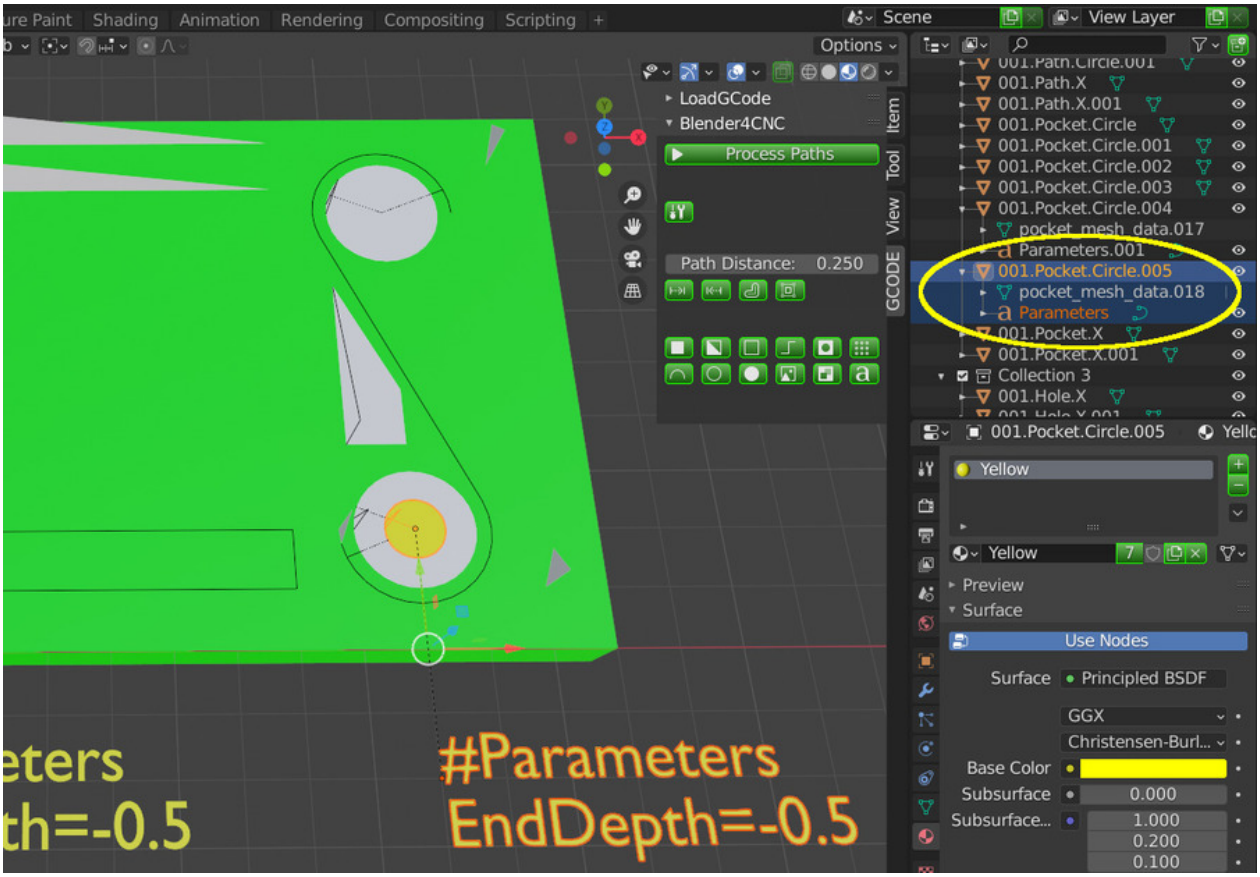

Figure 184: Take a moment to take a look at how the overridden parameters appear in the list to the right of the screen. When you click on the triangle beside an operation like the deep circular pockets, you can see that there is a parameters object listed underneath them. The parameters object can be called "Parameters" or "Parameters.001" etc. Just as long as the parameters object starts with the text "Parameters". You could for example, call the parameters object "Parameters.DeepPocket".

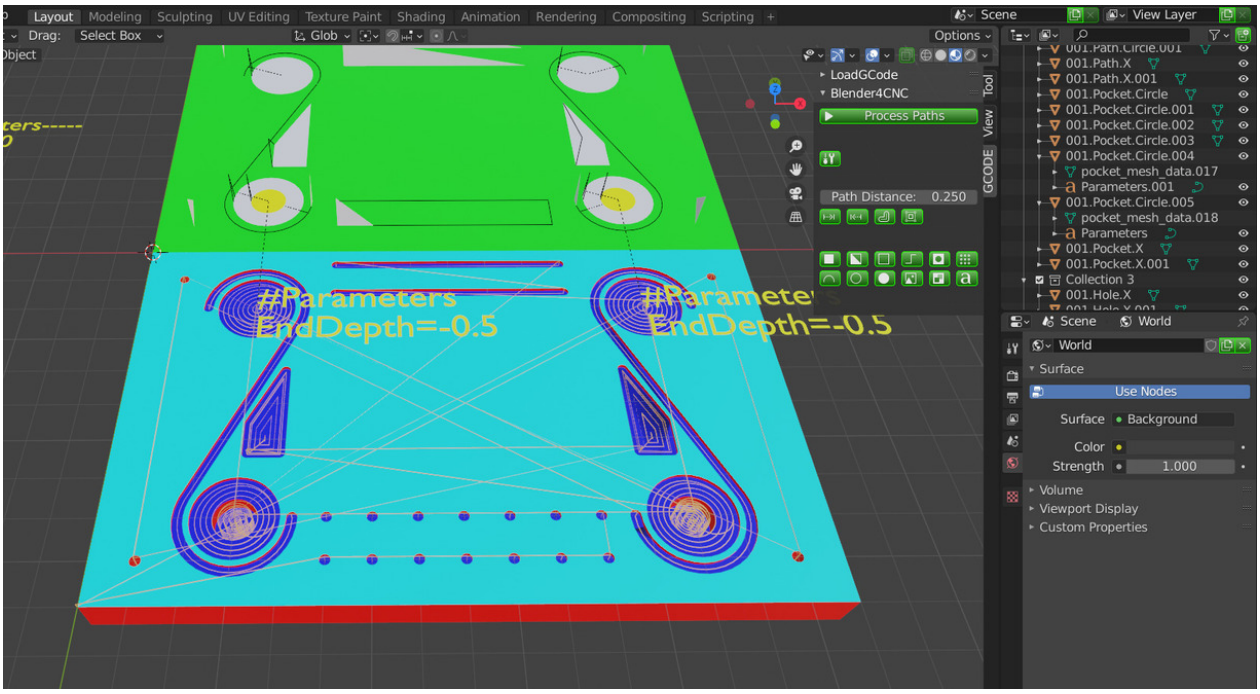

Figure 185: Click "Process Paths" and take a look at what we have accomplished so far. Notice how the path for the cutter is swirling and digging deeper into the last two deep pockets we created to achieve the greater depth.

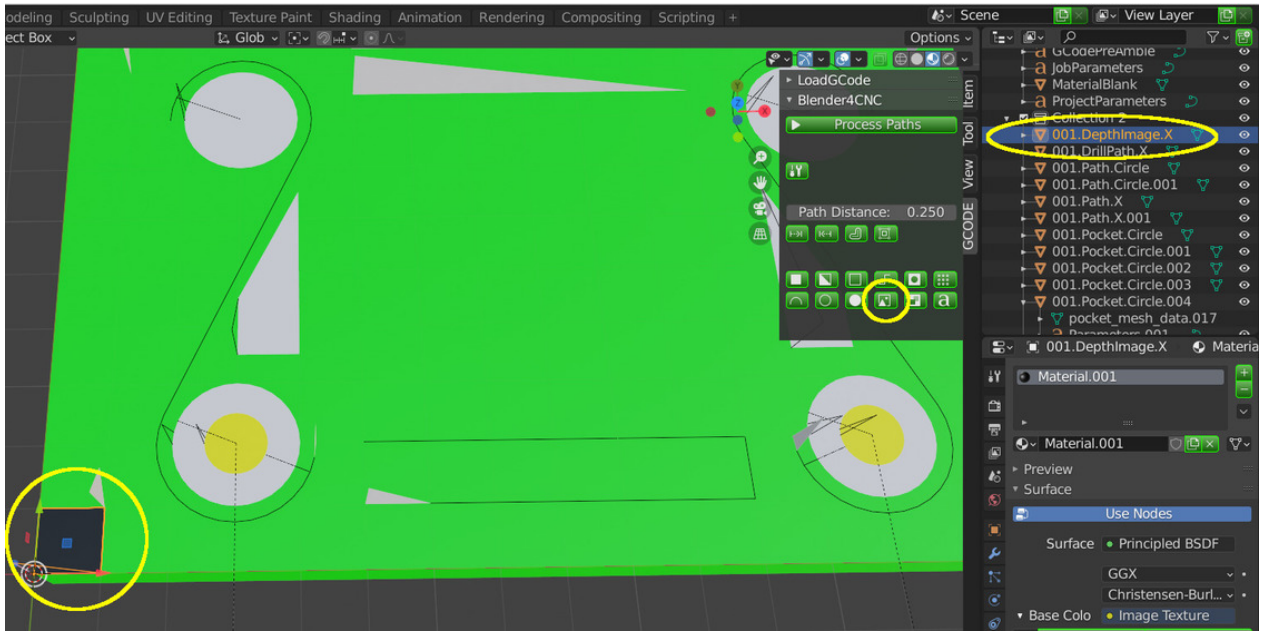

Figure 186: Let's now add a depth image in the middle of all the decorations. Click the "Create DepthImage" button.

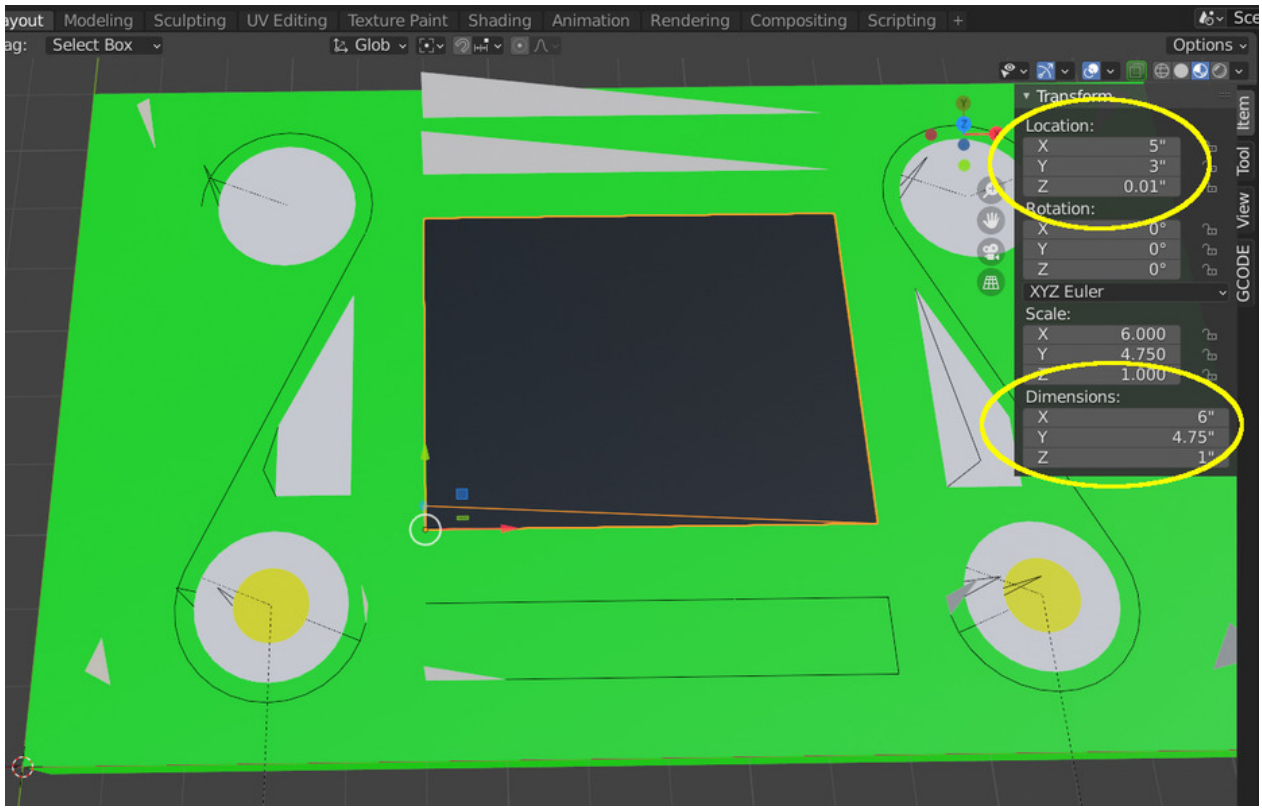

Figure 187: Move and resize the depth image operation.

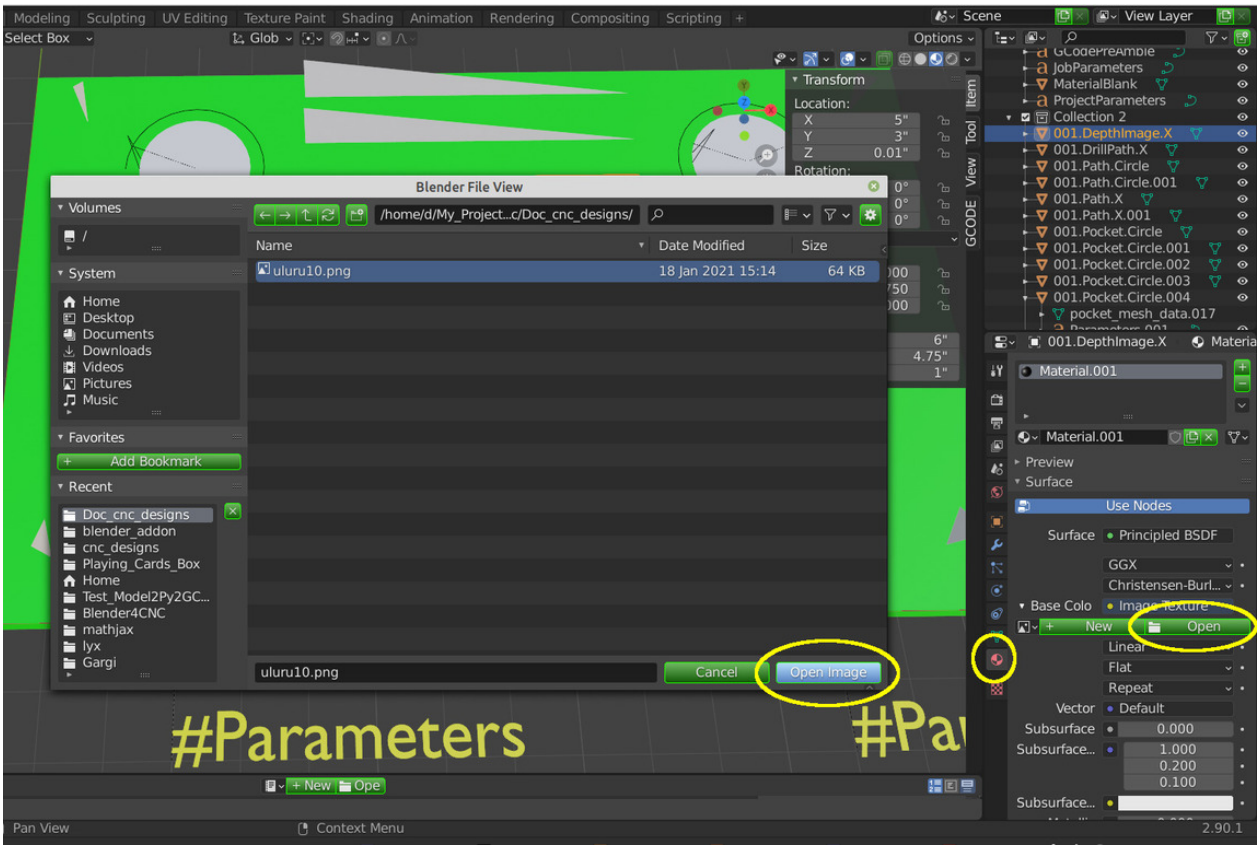

Figure 188: Click the "Material" tab and then click "Open" and browse to a grayscale depth image.

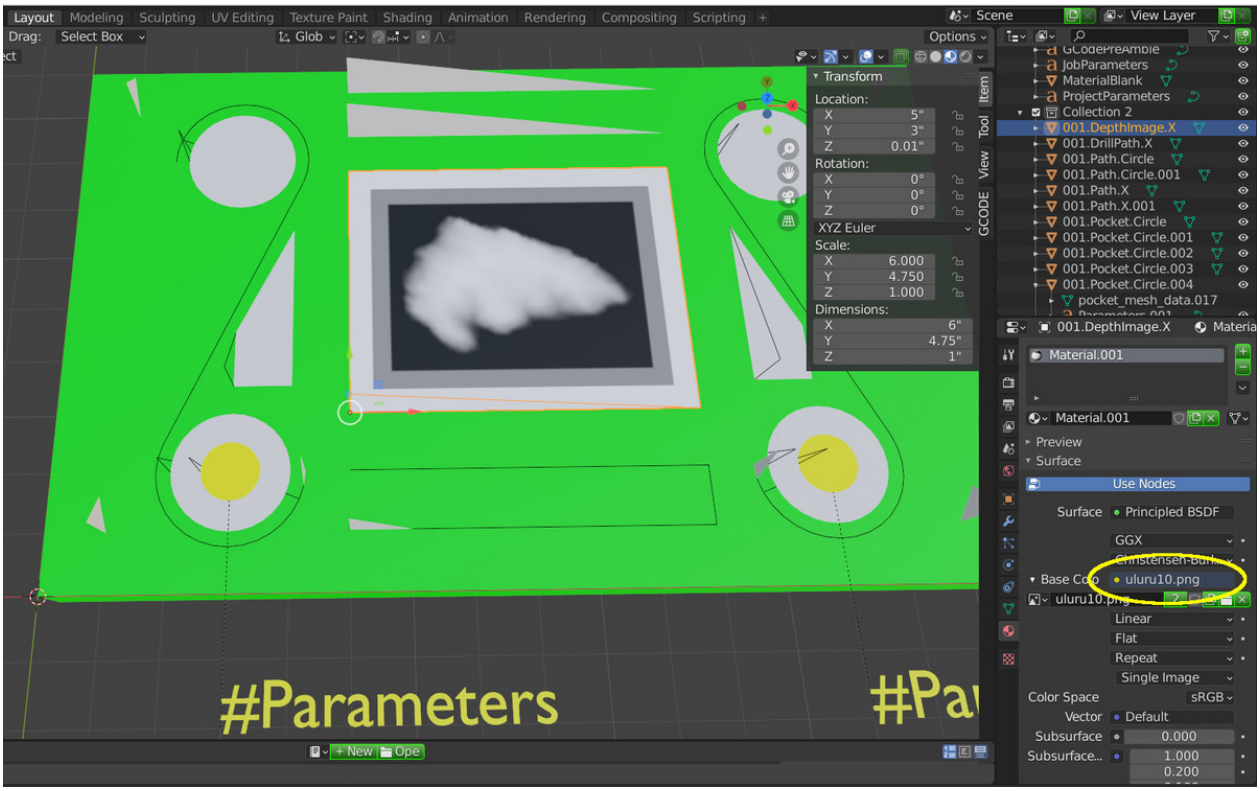

Figure 189: The image should appear in the viewport.

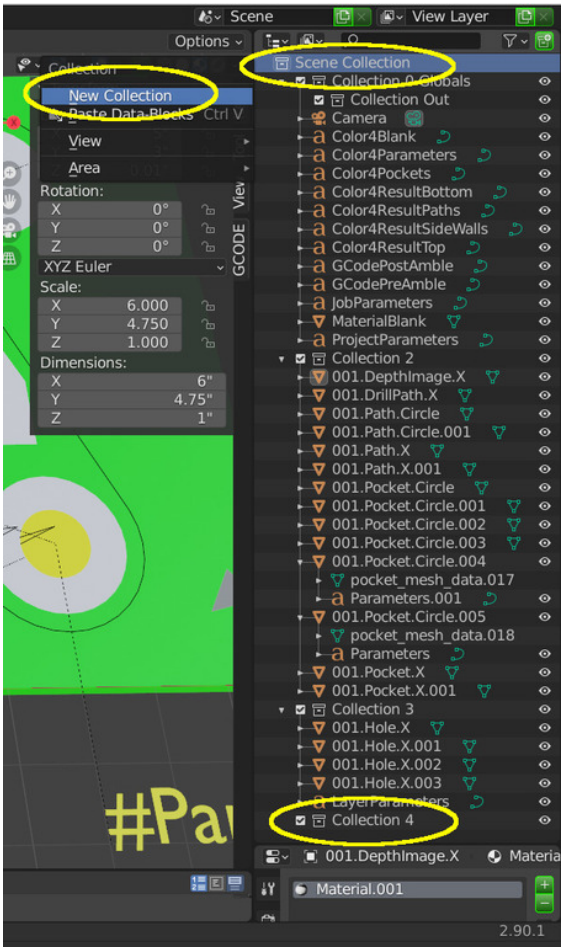

Figure 190: I want the image to be the last operation to be cut so let's put it in it's own collection and rename it with a high number. On the right side of the display, right click on "Scene Collection" and then "New Collection" and you should see a new collection called "Collection 4" appear.

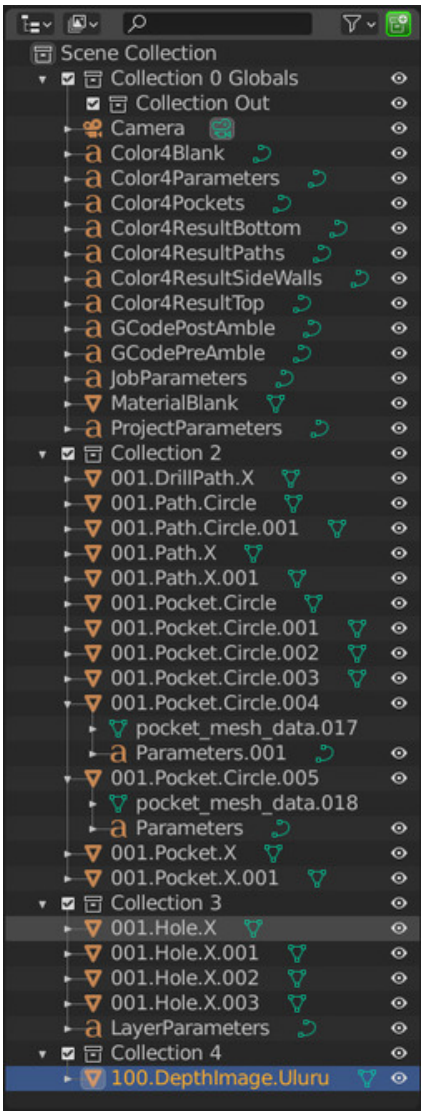

Figure 191: Drag and drop the depth image operation into "Collection 4" and then rename it to "100.DepthImage.Uluru" (double-click on the name text).

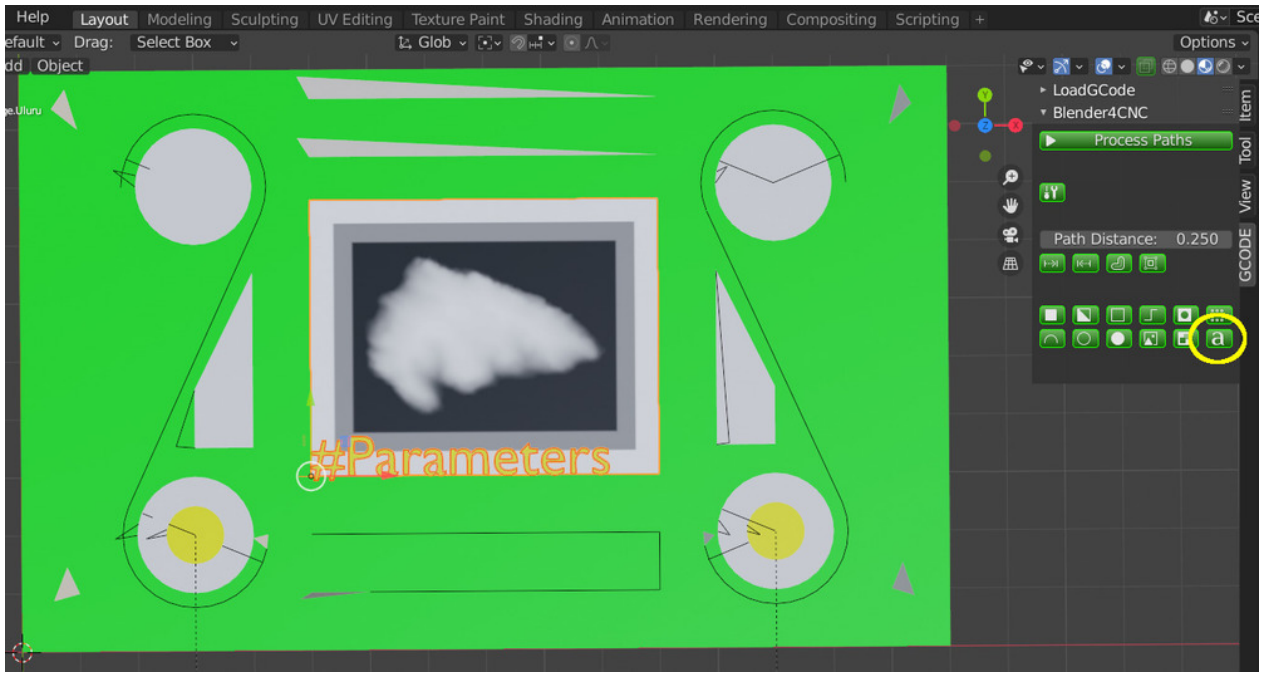

Figure 192: Add a parameters object to the depth image by selecting it and then clicking the "Create Parameters" button.

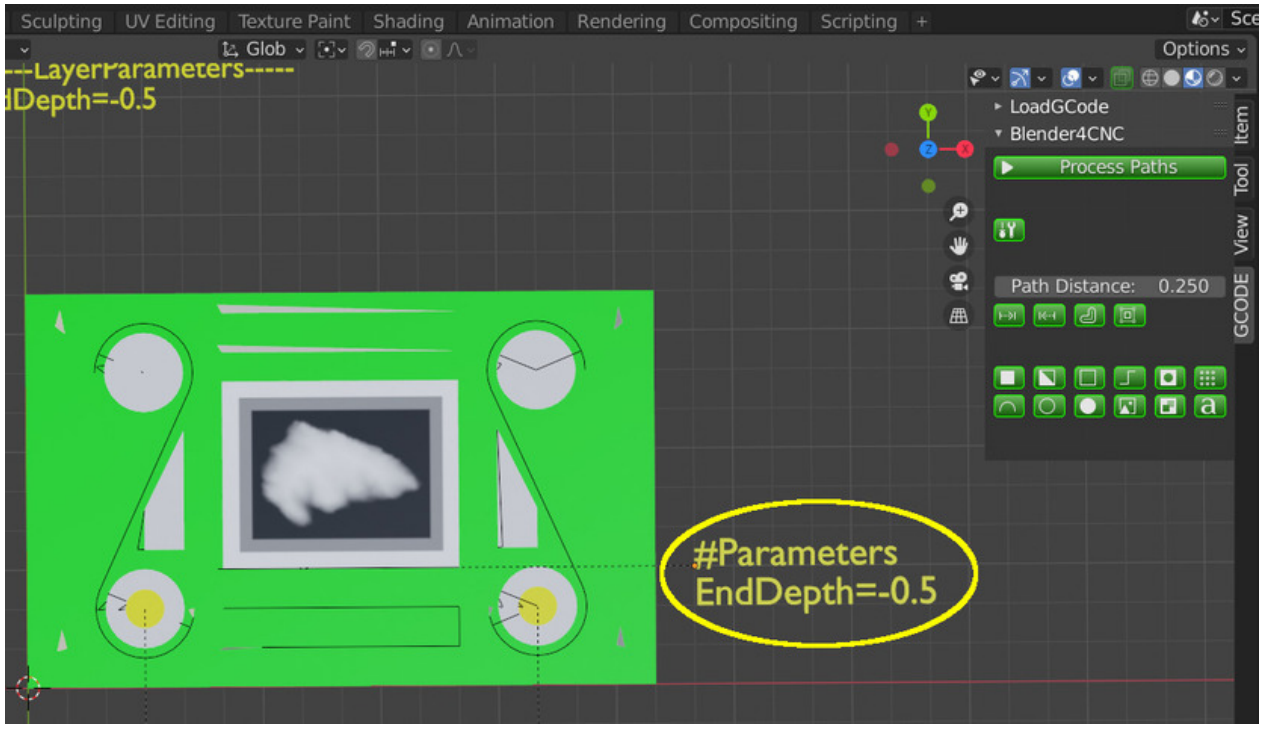

Figure 193: Change the end depth to -0.5".

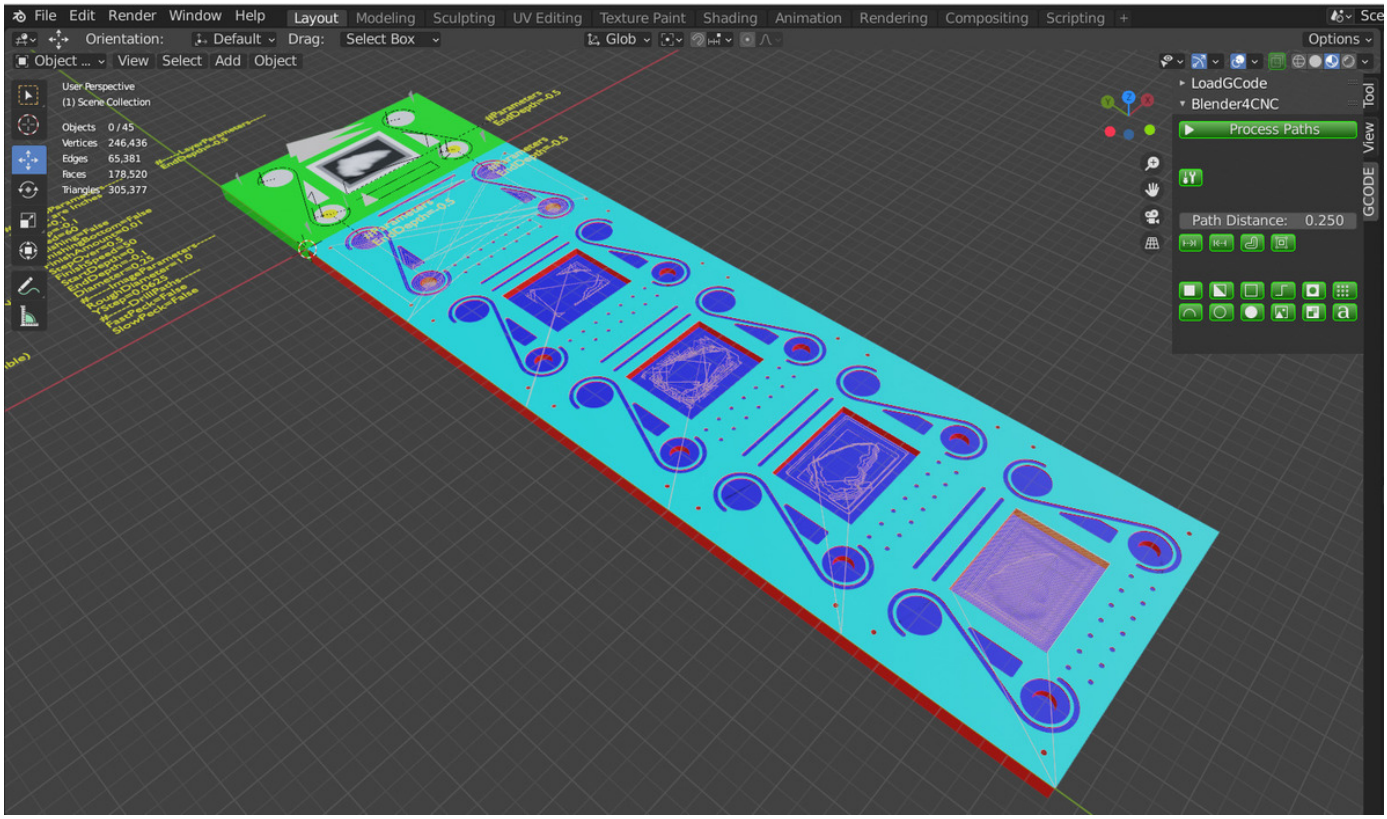

Figure 194: Click "Process Paths". See the section "The Multiple passes of a Depth Image" for more explanation about depth images.

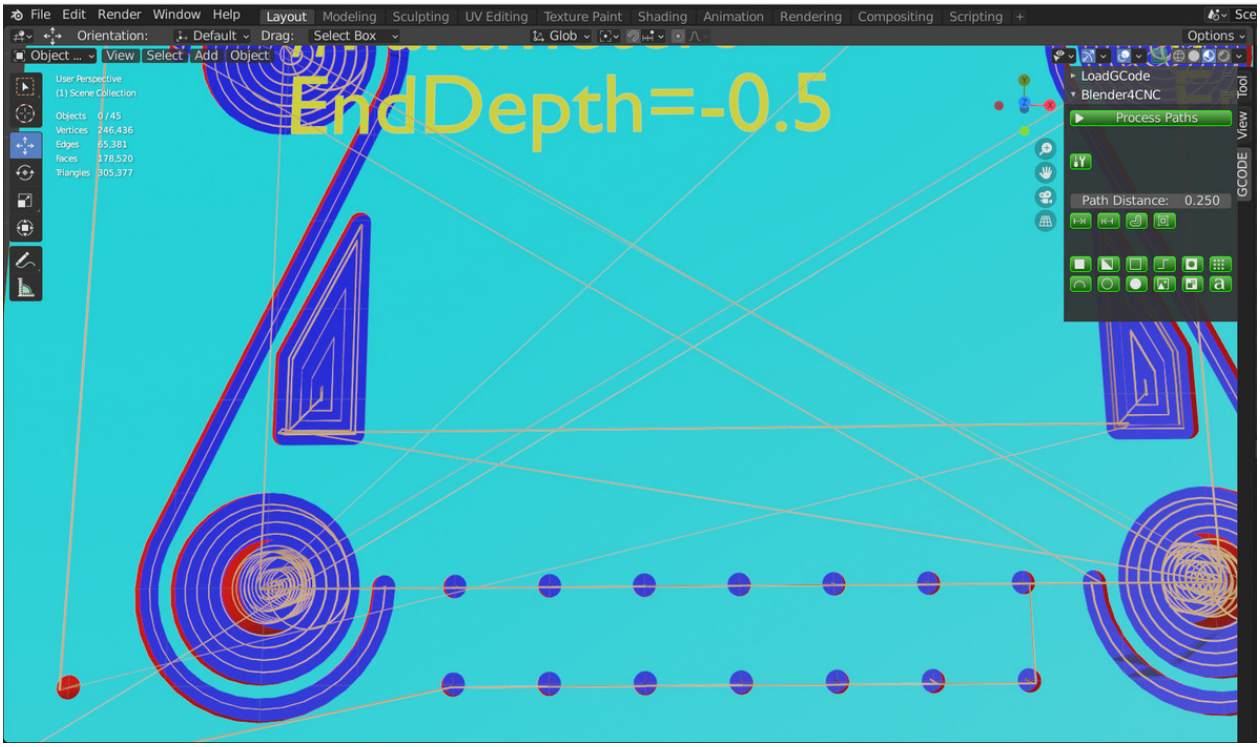

Figure 195: You can see on the first phase the path of the cutter is criss crossing across the project as it jumps from one operation to the next. Let's say we really want to minimize this type of unnecessary travel to save time when cutting the job. We can specify the order to cut the operations.

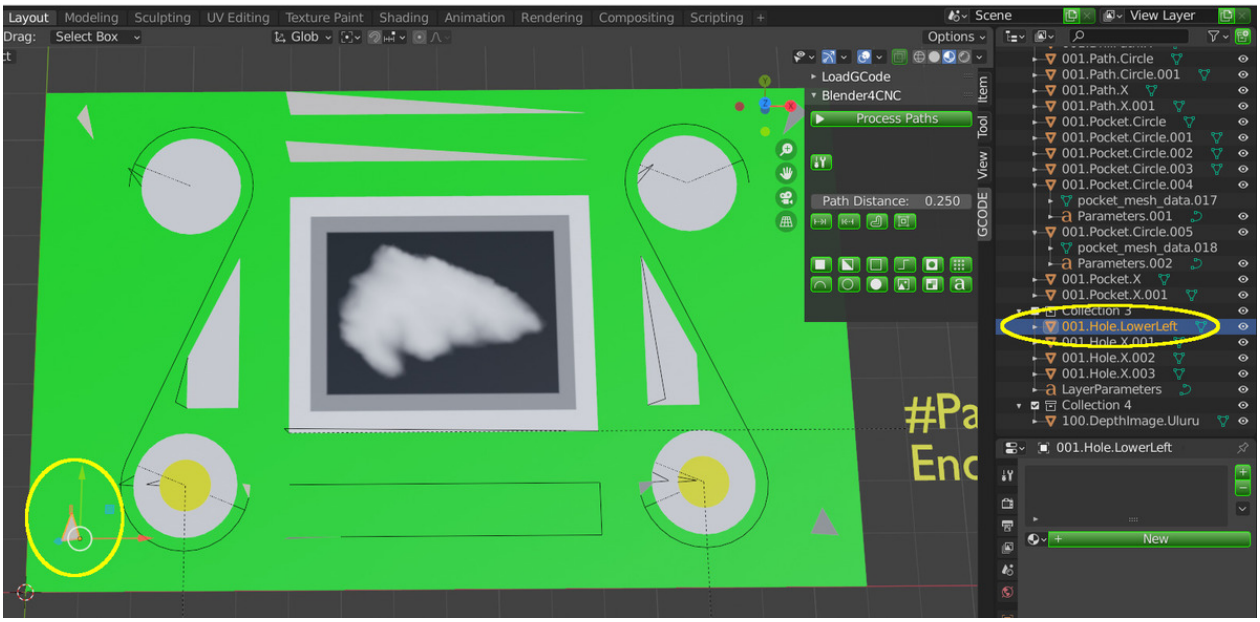

Figure 196: Let's first get really organized by providing helpful comments in the names of all our operatons. Begin by selecting the hole in the lower left of the project and rename it to "001.Hole.LowerLeft".

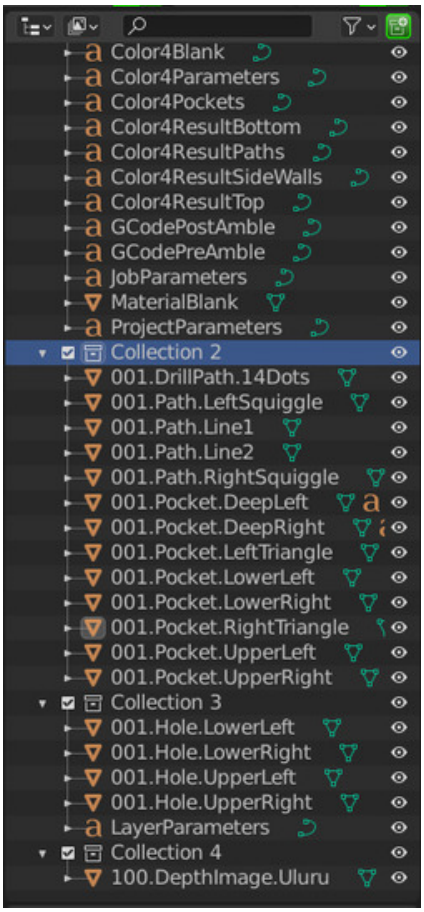

Figure 197: Select each operation one at a time and give each one a useful comment section in the name.

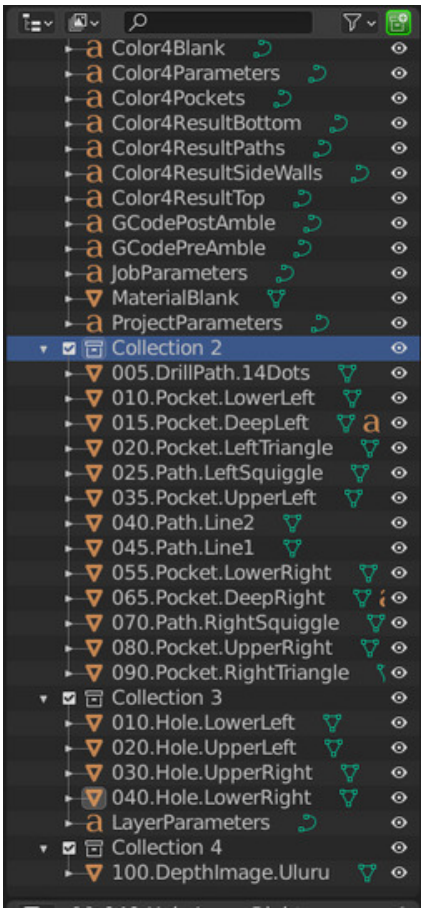

Figure 198: Now number each operation as shown. I like to step the numbers by 5 or 10 in case I come back later and decide I want to place an operation in between others. Blender4CNC always groups operations by collection so it will do all operations in "Collection 2" (in the numbered order in which they appear). Then it will do all the operations in "Collection 3" (in the numbered order in which they appear). Then it will do all the operations in "Collection 4" (in the numbered order).

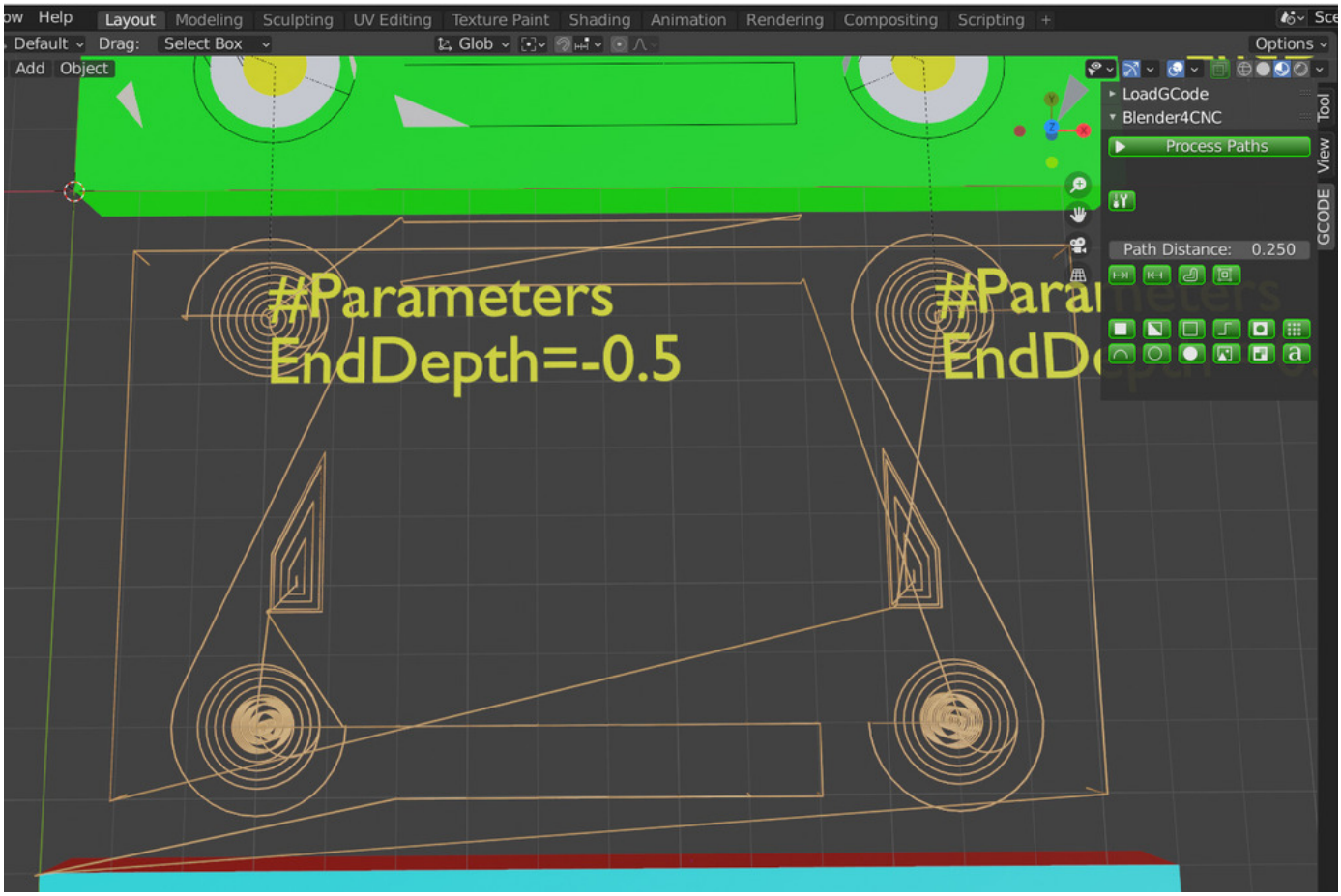

Figure 199: Once done, click "Process Paths" and you can see the criss-crossing has been minimized. (The material object was removed for clarity so just the path can be seen easily.)

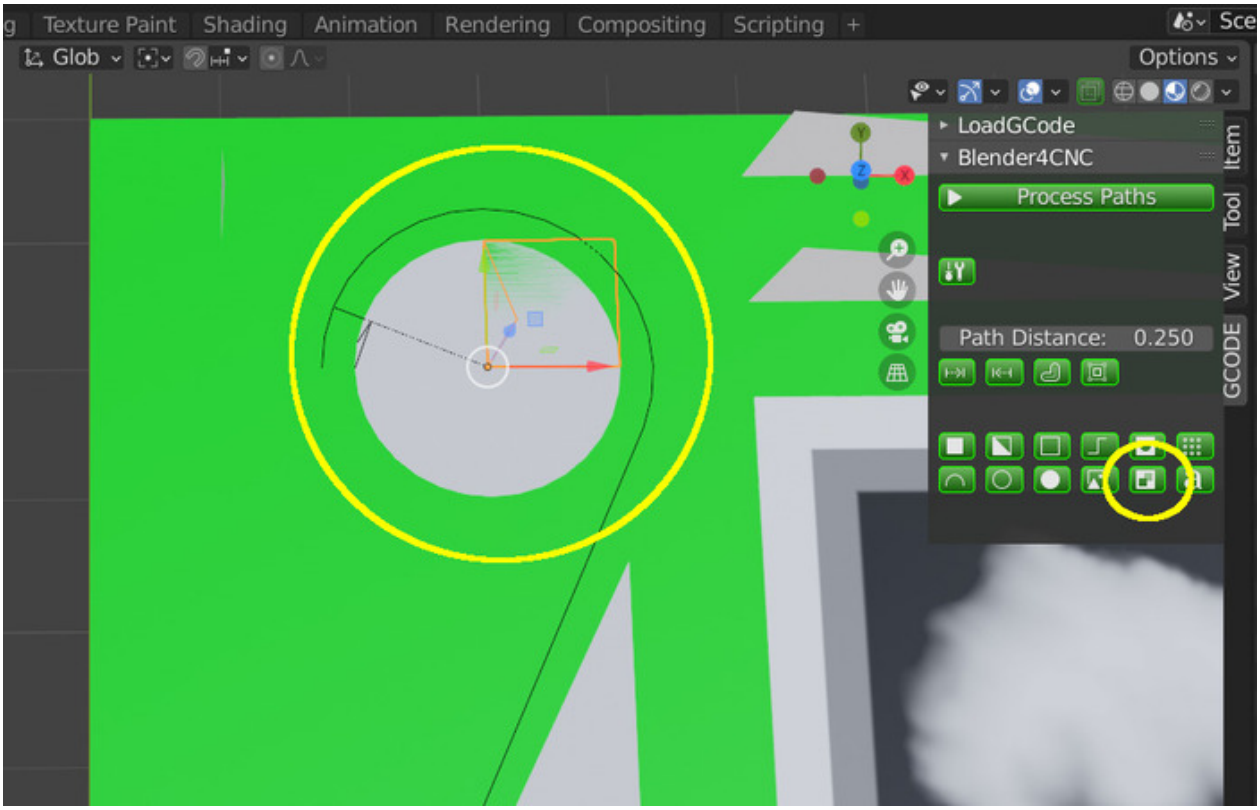

Figure 200: There is one type of operation that we have not used yet in this example - a tenon. Select the Upper Left circular pocket and click the "Create Tenon" button. A square tenon object will appear as a "child" object of the circular pocket.

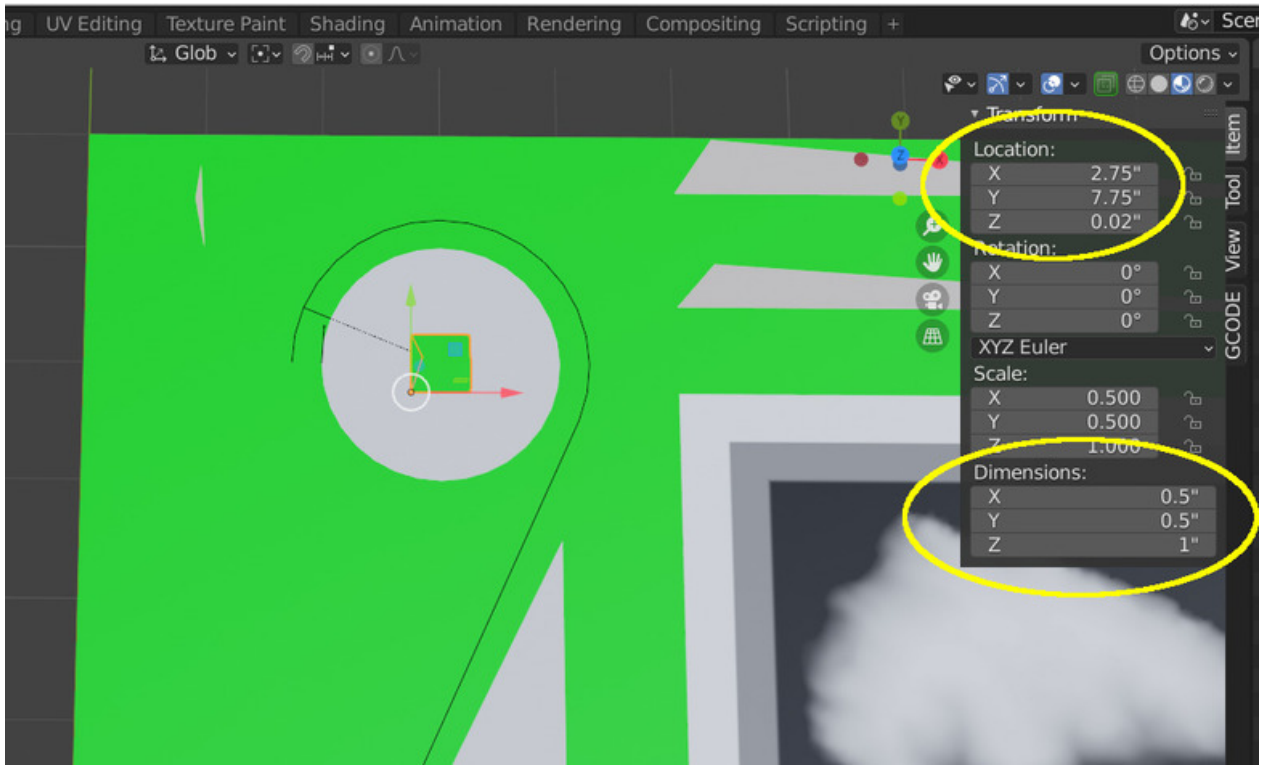

Figure 201: Change the dimensions and location of the new tenon (don't forget to set Z to 0.02 so it is nice and visible - if setting Z to 0.02" does NOT make it visible, then check if the gray pocket below is set to Z=0.01 - all operations except tenons should be added at a Z of 0.01" and it is possible that you moved or changed the Z level of an object). Now when the circular pocket is cut, this little square of material (an "island") should be left.

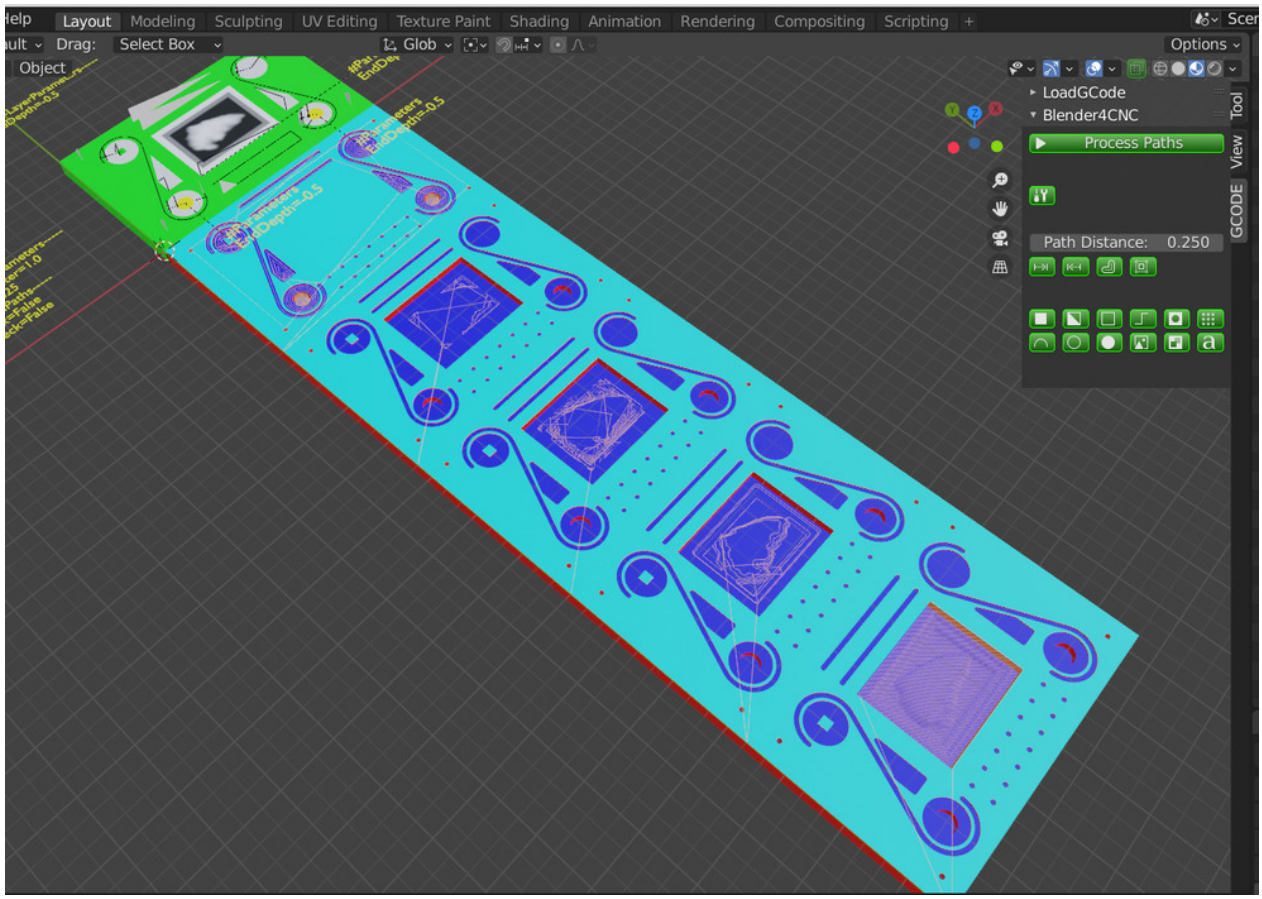

Figure 202: When "Process Paths" is clicked, we can now see that the tenon remains while the rest of the circular pocket is removed. This may take some time (not too bad on any modern computer) although you won't see any type of progress bar. It has been suggested that it show a progress bar while running - great idea! Unfortunately, Blender does not provide a way to do this for plugins that run for a long time (believe me, numerous developers have asked this question on the internet). If this ever becomes possible in the future, it will definitely be added! Also, a future version may speed up the process by using multiple cores to do the calculations so a progress bar may be less needed. If you have installed Blender in such a manner that it also opens a console/terminal window, then you can switch to that console/terminal window and see text messages scrolling by.

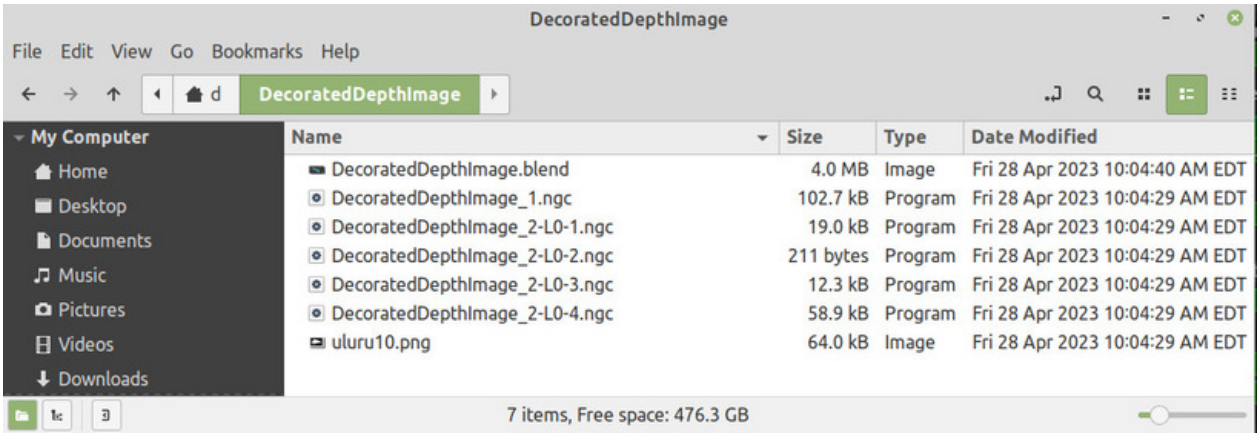

Figure 203: The GCode output files are in the same folder as the Blender project. The file "DecoratedDepthImage$_{1.ngc}$" contains all the GCode for all the operations except the depth image - these operations are all from collection 2 and 3 and are all done with a 1/4" flat cutter (if different cutters were used, this file would have been split up into multiple files). The remaining 4 ".ngc" files are for the 4 phases of carving the depth image. "DecoratedDepthImage$_2$-L0-1.ngc" is for roughing phase 1 with a 1/2" flat cutter; "DecoratedDepthImage$_2$-L0-2.ngc" is for roughing phase 2 and uses a 1/4" flat cutter; "DecoratedDepthImage$_2$-L0-3.ngc" is for roughing phase 3 and uses a 1/4" ballnose cutter and "DecoratedDepthImage$_2$-L0-4.ngc" is for the final carving phase and uses a 1/4" ballnose cutter.

1.5 Some Blender Tips

Tip #1 - If you have lost sight of your model (i.e. you have panned far away and all you have is an empty screen and no way of knowing which way to pan to get back) press the "Home" key to center all your objects in the screen.

1.5.1 View Vertex Indices

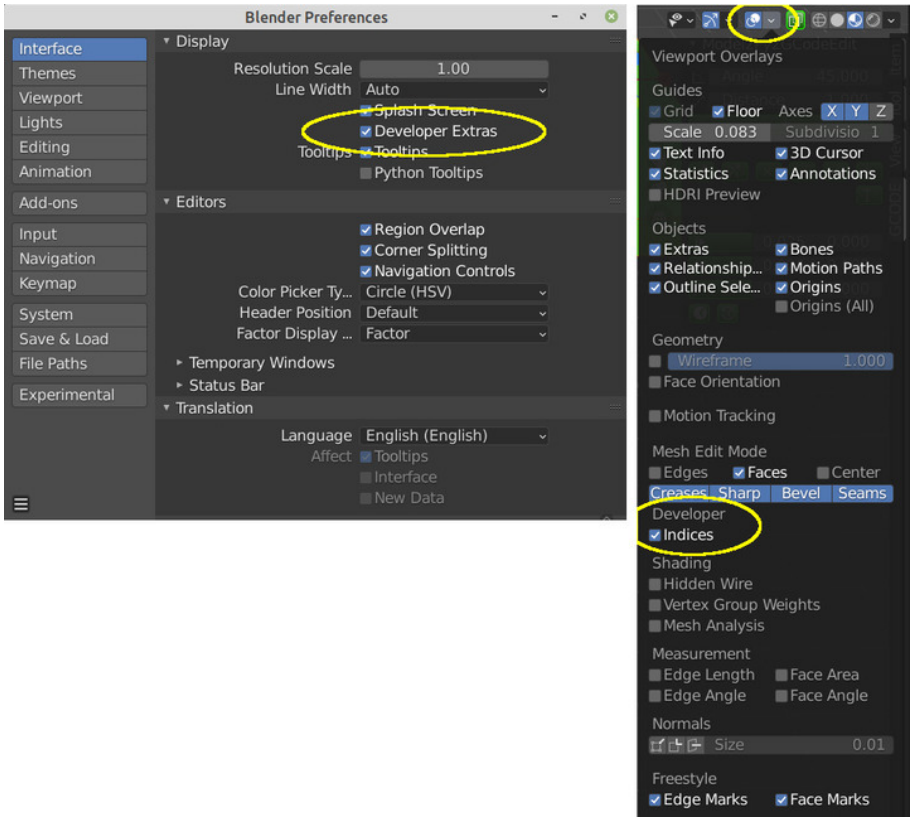

Figure 204: To see the numbering of vertices in Edit Mode go to Preferences, Interface, and check the "Developer Extras" box. Then select a mesh and enter Edit Mode. Then select the "Viewport Overlays" button and check the box for "Indices" under "Developer".

1.5.2 View Clipping Planes

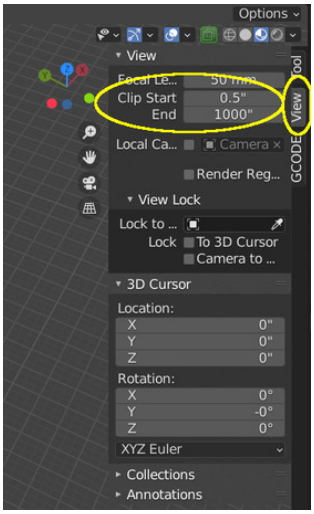

Figure 205: Sometimes you lose sight of an object if you zoom in too far. This happens if the object is closer than the "Start Clip" plane. Likewise, sometimes you lose sight of the whole scene (or parts of it) if you zoom out too far and this is because the whole scene (or at least parts of it) are beyond the "End Clip" plane. If this happens a lot, Go to the "View" tab on the right side of the 3D viewport and change the "Clip Start" and "Clip End" values to suit your needs.

1.5.3 3D Cursor Location

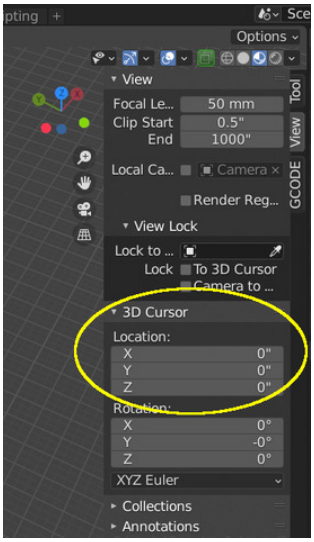

Figure 206: If the 3D cursor is in a weird location you can move it back to 0,0,0 by going to the View tab on the right side of the 3D viewport.

1.5.4 Viewing and Zooming

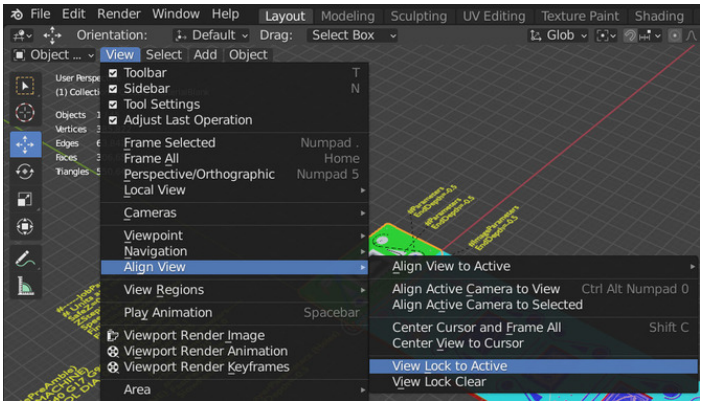

Figure 207: Sometimes zooming and scrolling around the project is not quite working well for the object you are interested in. To help make Blender zoom on the object of interest, you may want to "lock" the view to the active object. Select the object of interest (probably already done so if you're reading this) and go to the "View" menu -> "Align View" -> "View Lock to Active". If you have lost sight of your model (i.e. you have panned far away and all you have is an empty screen and no way of knowing which way to pan to get back) press the "Home" key to center all your objects in the screen.

1.5.5 Setting Multiple Object to same Z height

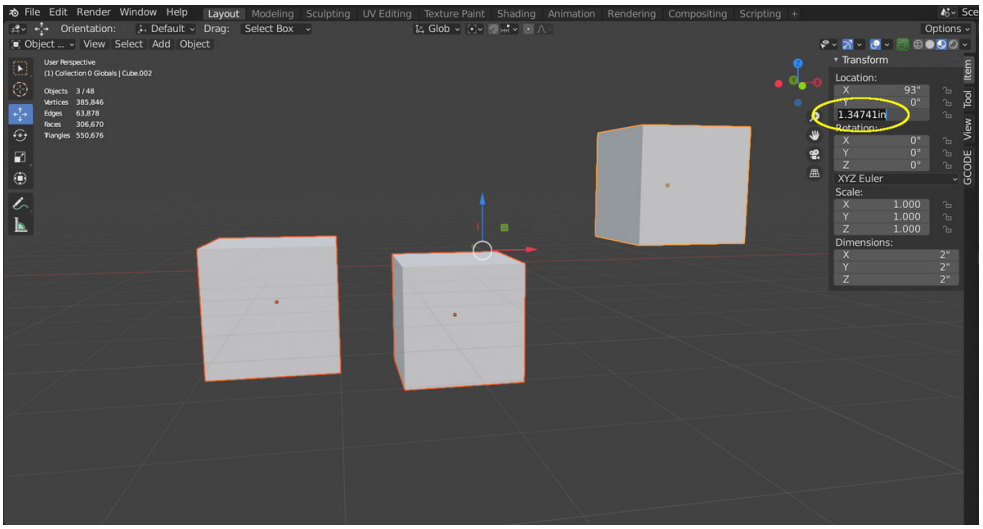

Figure 208: To align multiple objects to a specific Z height (this works for X and Y too), select all objects and then set the Z value. For example to align all three cubes to 0, set the Z to 0. At first, you will see only the "active" object (the last one selected) is moved.

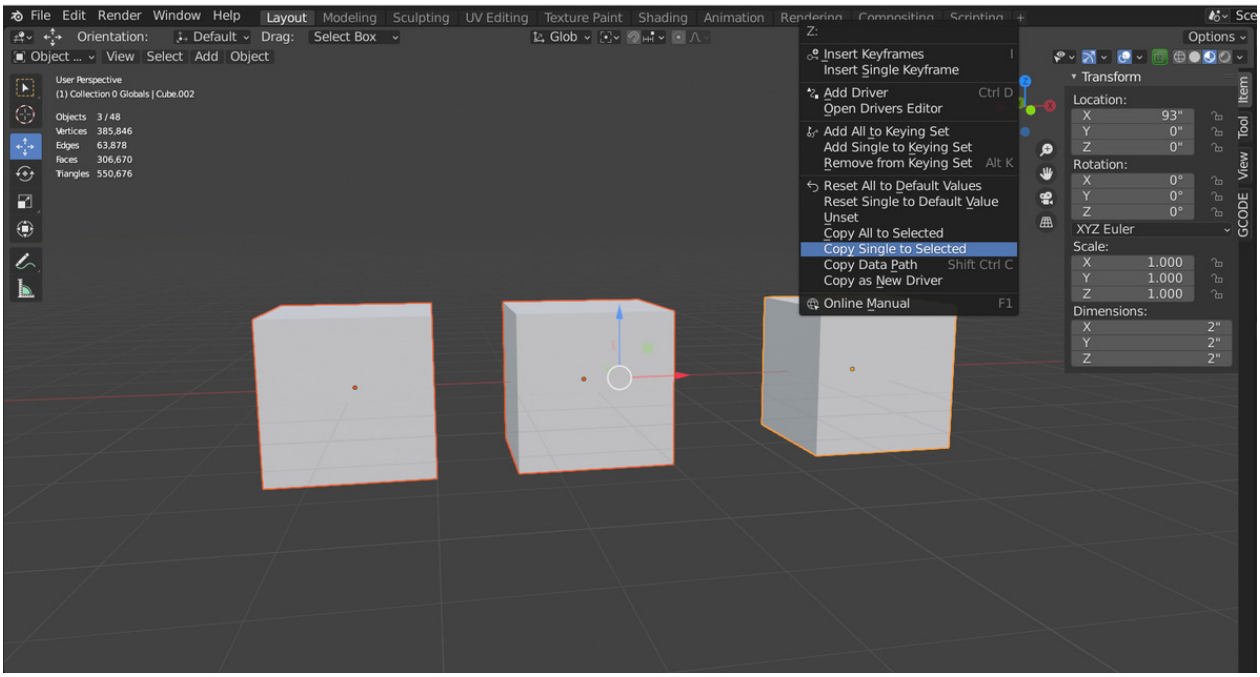

Figure 209: Right click on the "Z" field" and select "Copy Single to Selected" and you will see that value get applied to the other selected objects.

1.5.6 Setting Text Size

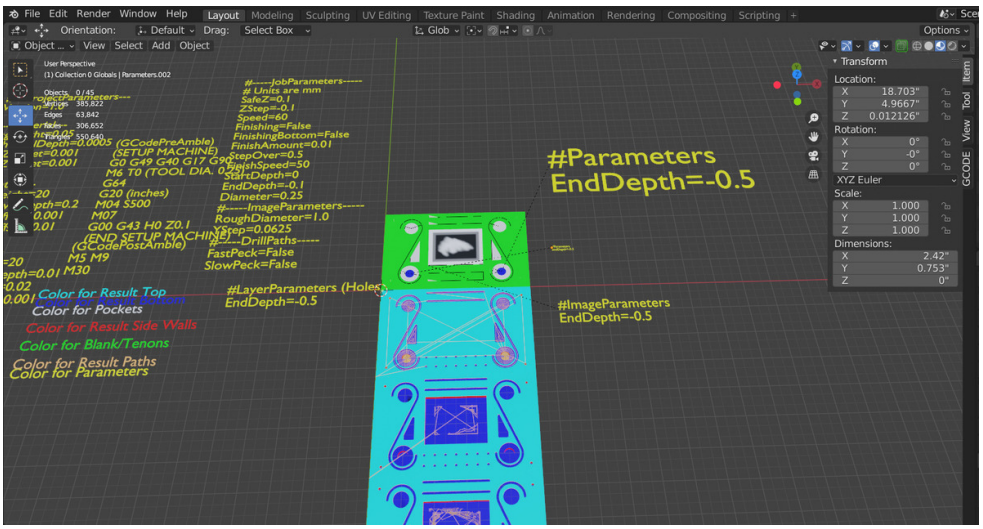

Figure 210: Sometimes different text objects get smaller or larger because they are "children" of an attached "parent" object which got scaled. There is a way to set all text to be the same size.

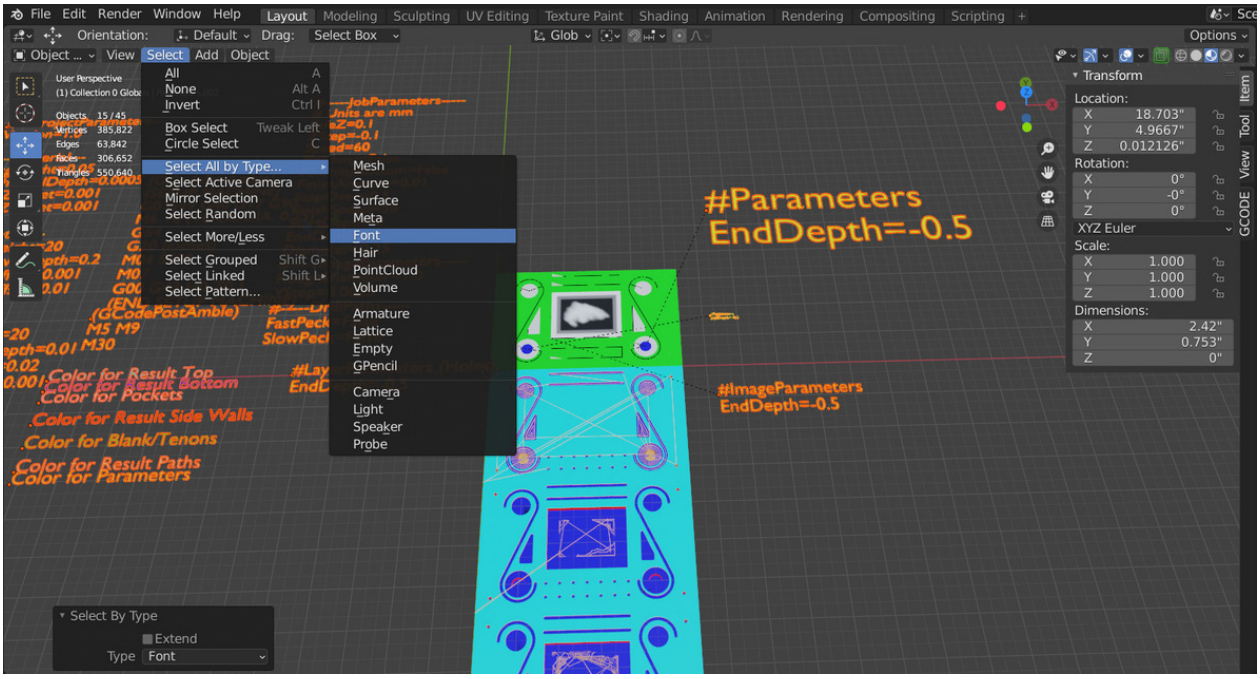

Figure 211: Select all text objects by going to the "Select" menu and then clicking "Select All by Type..." and then "Font".

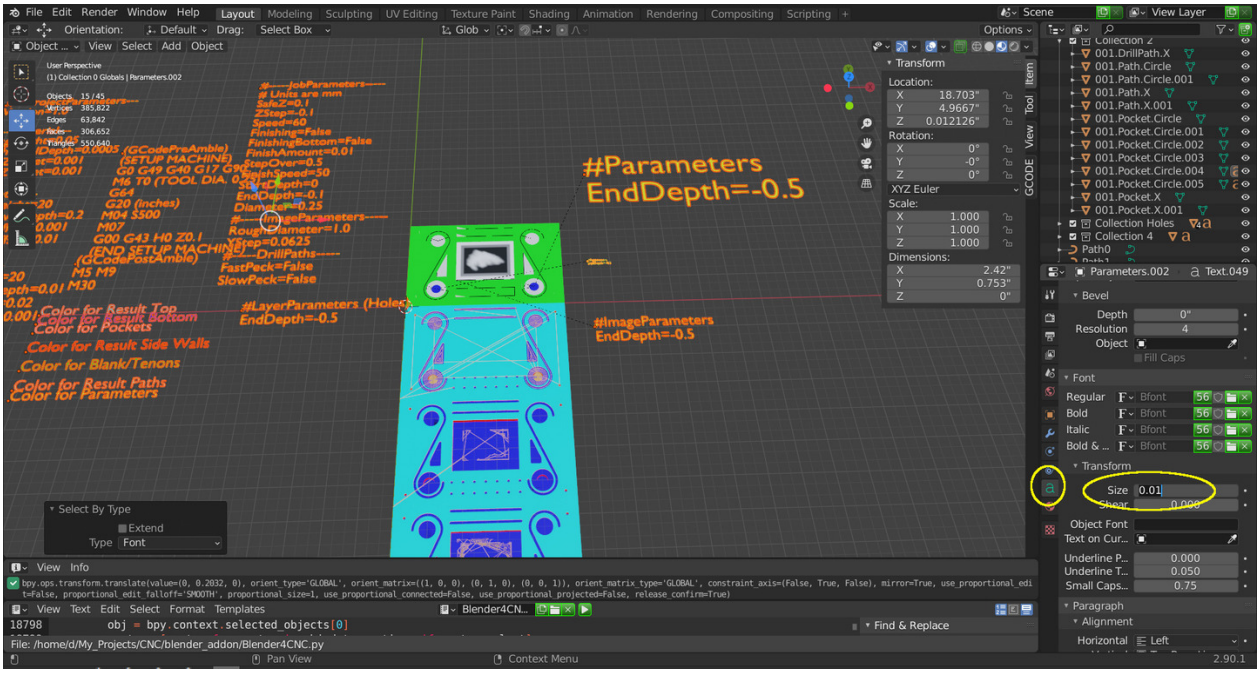

Figure 212: On the "Font" tab on the right side of the screen, you can change the "Size" parameter. Initially, only the "active" (last selected) object will be changed.

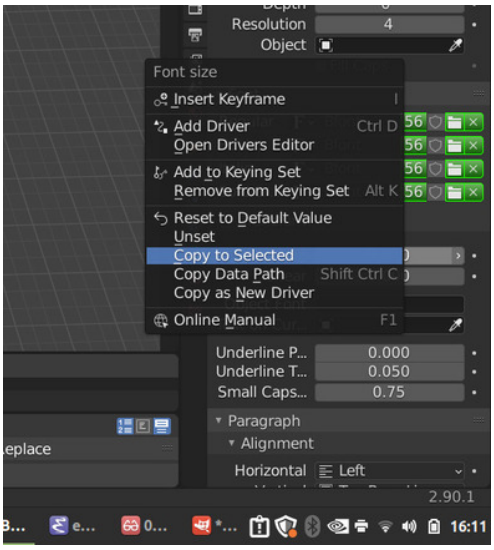

Figure 213: Right click on the "Size" field and select "Copy to Selected" and the size value will be applied to all other selected text objects - they should all look the same size.

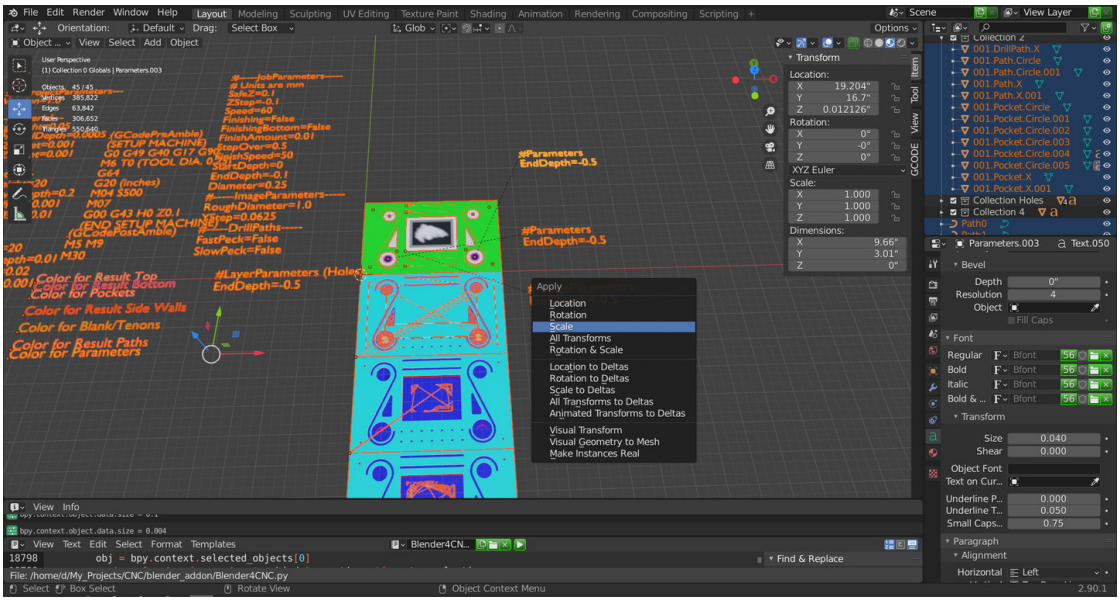

Figure 214: Note that sizing text objects is affected by any "scaling". It can be wise to "apply the scale" to all objects in the scene that have been modified a lot before setting the text size. Press "a" to select all objects in the scene and then press ctrl-a to bring up the "Apply" menu and then select "Scale".

1.6 Parameters

1.6.1 Job Parameters

Table 3: Table of Parameters. The examples are for a machine that is imperial (inches).

Name	Description	Example
	The following parameters are relevant for the whole job:	
SafeZ	A safe Z height where the tool will not hit the material when performing rapid movements (G0 codes)	0.1
ZStep	When cutting deep pockets or paths, the generated GCode will step in the Z axis by this amount for each depth pass.	-0.1
Speed	Rapid and movement speed.	60
Finishing	Whether or not to perform a "finishing" pass on pockets. A pass is a slower pass around the edges of a pocket that will remove a **very** tiny amount. When a "finishing" pass is used, it will calculate paths such that the pocket will be cut to be smaller by "FinishAmount" amount - leaving exactly that tiny amount of material to be removed.	True
FinishingBottom	Whether or not to perform a "finishing" pass on the bottom of the pocket.	False
FinishAmount	The amount to "finish" in a finishing pass.	0.01
StepOver	The percentage of bit overlap when pocketing (as a value between 0 and 1 where 0 = 0% and 1 = 100%)	0.5
FinishSpeed	The movement speed when finishing	50
StartDepth	The Z height to start pocketing (if a smaller, deeper pocket is being cut within a larger pocket, it can start at the depth of the initial pocket to save time).	0
EndDepth	The final depth of a pocket.	-0.5
Diameter	The diameter of the router bit.	0.25
OutputFilename	The name to be given to any output GCode files.	"ABC"

Table 4: Table of Parameters (cont'd). The examples are for a machine that is imperial (inches).

Name	Description	Example
	The following parameters are specific to depth images:	
RoughDiameter	The diameter of the router bit to be used for the first rough pass. Usually large to remove a lot of material quickly. Only the first stage of cutting a depth image will use this router bit.	1.0
YStep	The distance between "lines" in a depth image. A depth image is produced by cutting many lines very close to each other to produce a smooth result.	0.0625
	The following parameters are specific to drill paths:	
FastPeck	Holes can be plunged or "pecked". Fast pecking specifies to peck down and pull the bit up a little bit before going down a little further	False
SlowPeck	Slow pecking specifies to peck down and then fully retract the bit from the hole before pecking a little further down.	False
PeckDepth	The amount to drill on each "peck" if using slow or fast peck drilling	False

Note, if the "OutputFilename" parameter is not specified in the JobParameters or in any "LayerParameters" objects within any collections, then the project name will be used to name all output GCode files. For example, if the project name is "abc.blend", then output GCode files will be called "abc-1.ngc", "abc-2.ngc" etc. See the section "Output Filename Parameters for more explanation of how to organize projects and provide useful names for the output GCode files.

1.6.2 Job Parameters that affect each Operation

Table 5: "X" indicates the parameter affects the operation.

Parameter	Pocket	Path	Hole	Drill Path	Depth Image
SafeZ	X	X	X	X	X
ZStep	X	X			X (Rough Pass)
Speed	X	X	X	X	X
Finishing	X				
FinishingBottom	X				
FinishAmount	X				
StepOver	X				
FinishSpeed	X				
StartDepth	X	X			
EndDepth	X	X			X
Diameter	X	X	X	X	X (Final Pass)
OutputFilename					
RoughDiameter					X (Rough pass)
YStep					X (Final Pass)
FastPeck				X	
SlowPeck				X	
PeckDepth				X	

1.6.3 Overriding Parameters

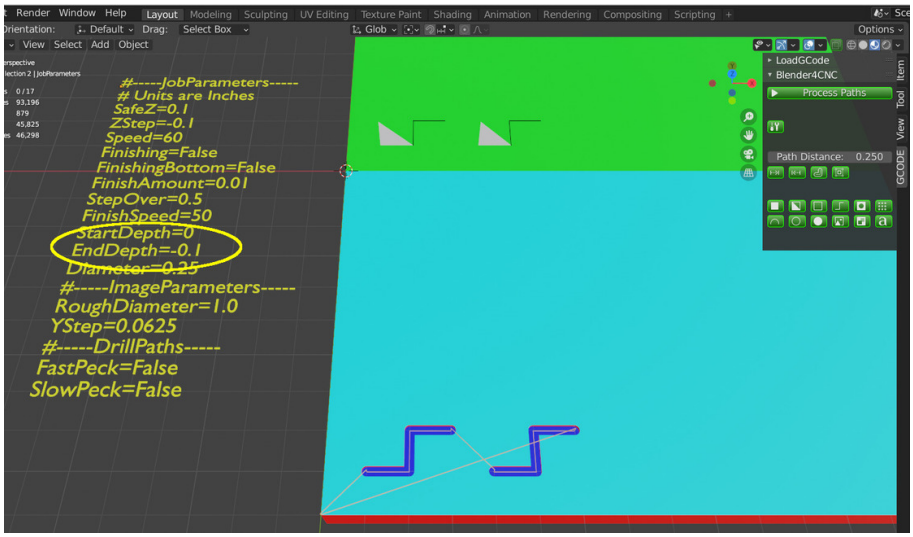

Figure 215: The JobParameters text defines the job. But these are only the default parameters to use when parameters are not explicitly set for an operation. You can override these defaults by explicitly setting parameters on an operation. Here we have a project with two path operations. They look identical when cut because they both use the default parameters with a bit diameter of 0.25"

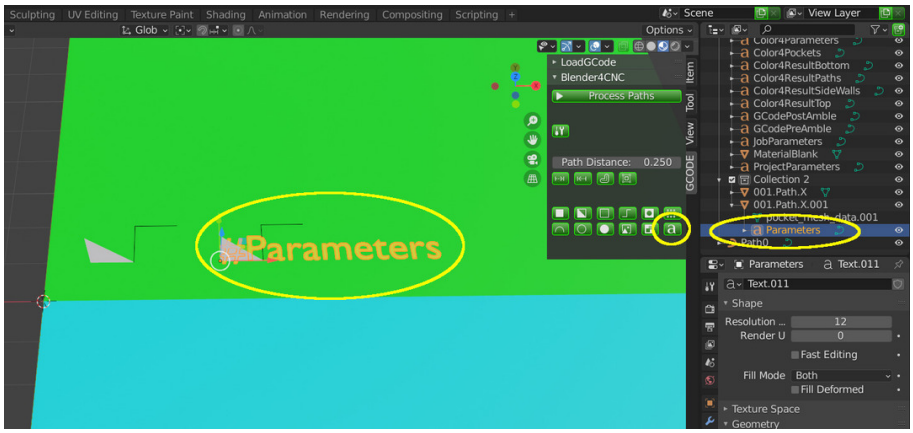

Figure 216: Select the path on the right and click the "Create Parameters" button and you will see a new text object added to the scene. The new text object is called "Parameters" (or maybe "Parameters.001, etc.) and is a child of the selected operation.

Figure 217: If you move the parameters object around you can see that it is connected to the parent object by a black dotted line

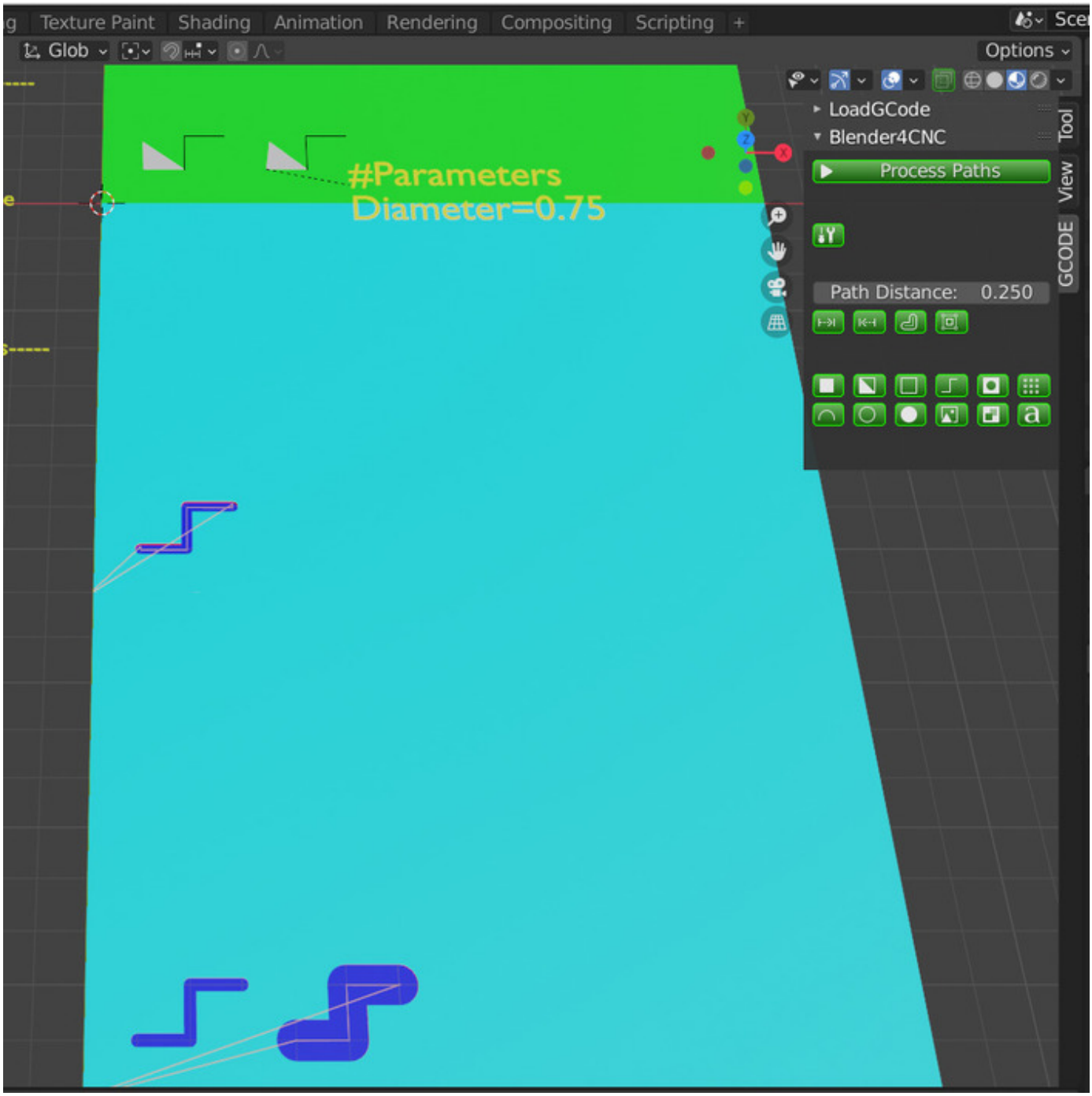

Figure 218: Edit the parameters (press TAB) and add a line "Diameter=0.75" then press TAB again. Click "Process Parameters" and you will see that the path on the right is now much wider as it is being cut with a 0.75" bit instead of 0.25". Notice also that the project now uses two different bits. Therefore, the operations are visualized in two phases (the job now requires a tool change).

1.6.4 Parameters specific to a Collection

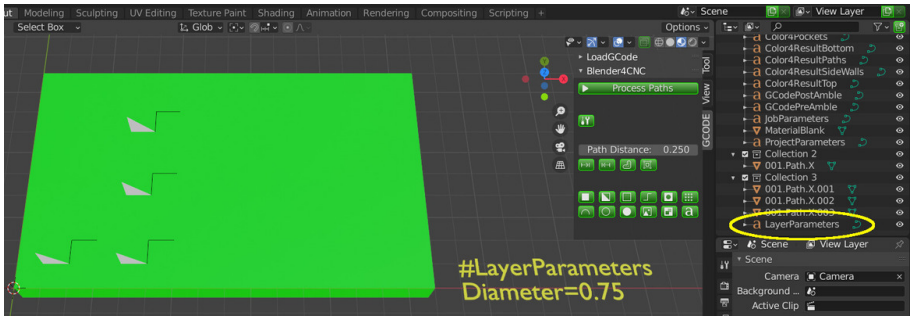

Figure 219: It is also possible to override default parameters by creating parameters in a collection. Here is an example with 4 path operations. Three of the operations are in "Collection 3". Parameters in a collection **must** be called "LayerParameters". While the default job parameters specify a bit diameter of 0.25" for this project, "Collection 3" has a "LayerParameters" object which overrides the diameter to be 0.5". This is more convenient than attaching a parameters object to each of the three paths. So a benefit of organizing operations by collection allows you to set parameters once in the collection.

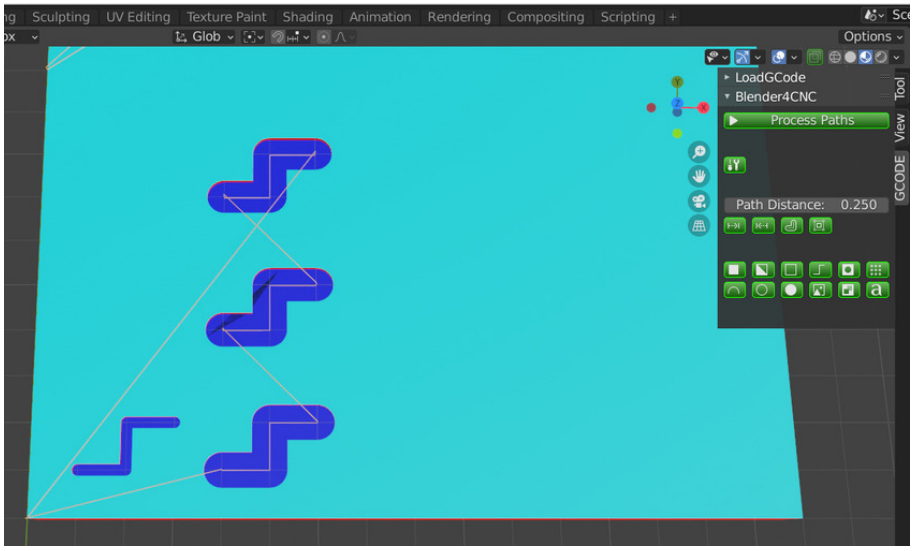

Figure 220: When "Process Path" is clicked the operations are visualized and here we have zoomed in on the final phase. The first operation is cut at 0.25" and the next 3 paths are cut with a 0.5" bit. So there are 3 ways to specify parameters for operations in a project - (1) in the JobParameters object at the project level, (2) in a LayerParameters object in a collection or (3) in a single Parameters object attached to an individual operation.

1.6.5 Output Filename Parameters

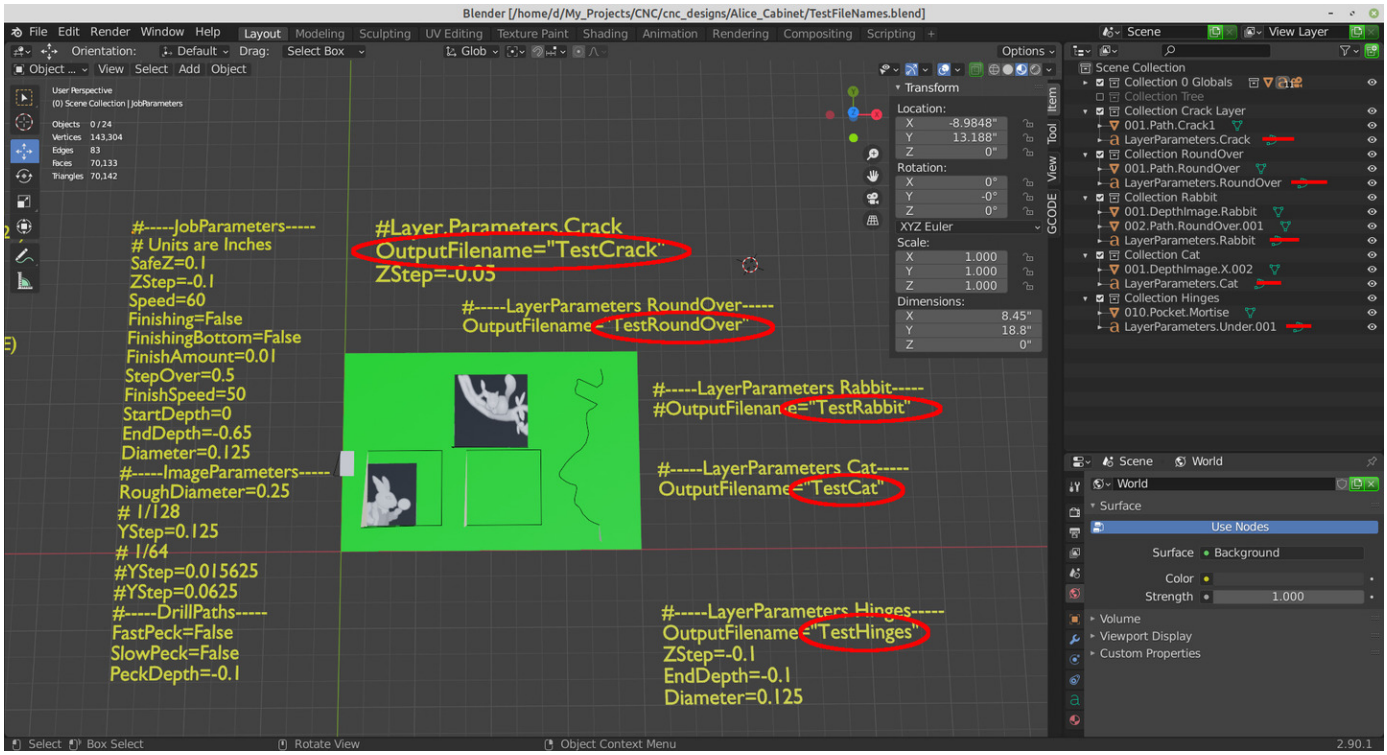

Figure 221: It is possible to organize a multi-part project by providing a filename in the layer parameters in specific collections within a project. Here is a project which has been organized into 5 collections. Each collection has a text object named "LayerParameters..." and each of these has defined the variable "OutputFilename" to give a preferred output name to each collection of operations.

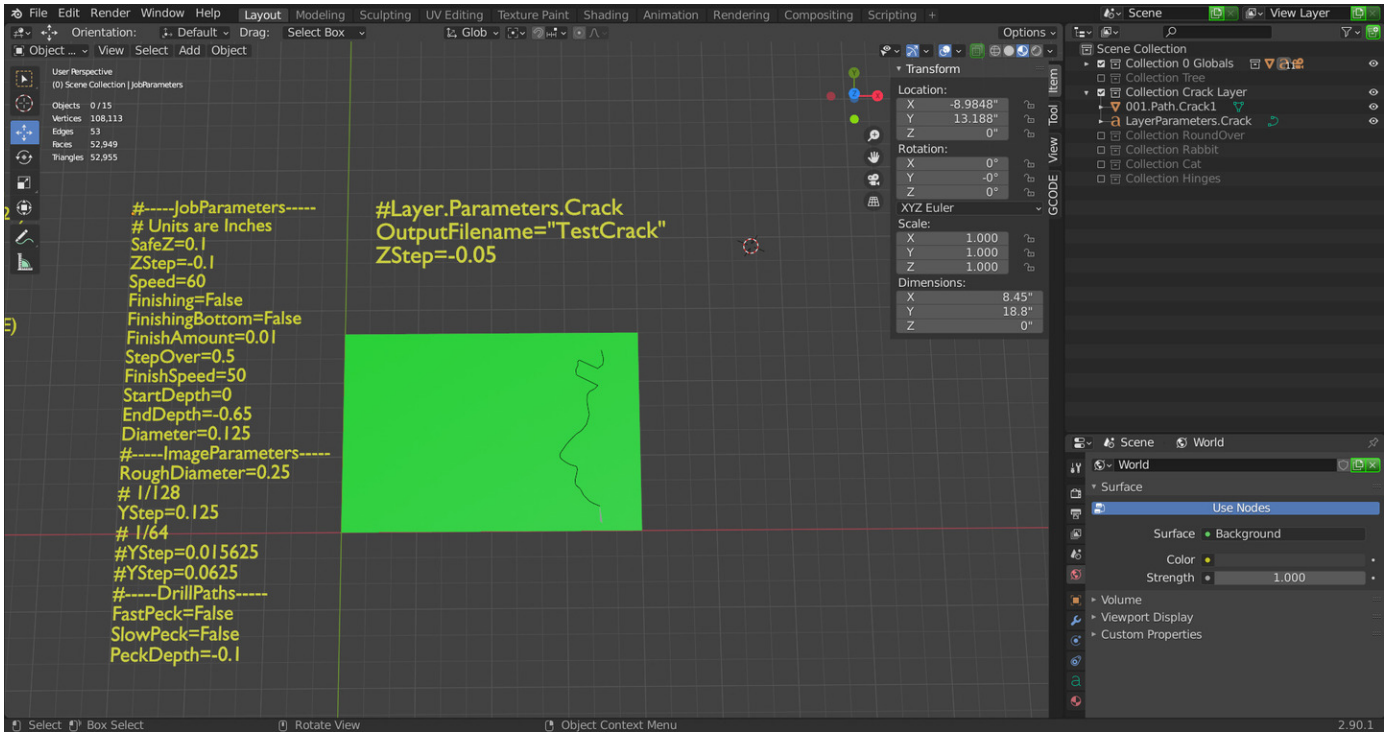

Figure 222: The collection "Collection Crack Layer" draws a squiggly line that looks like a crack.

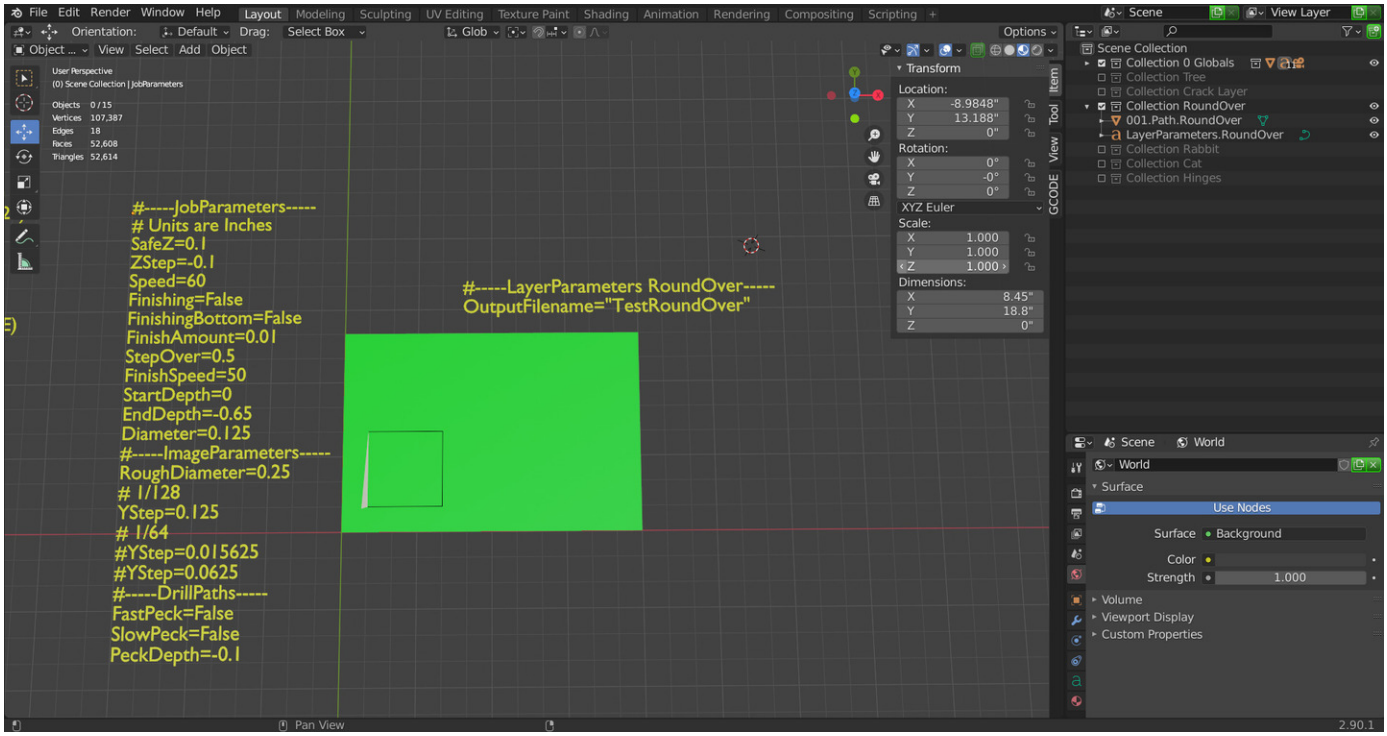

Figure 223: The collection "Collection RoundOver" draws a rectangle.

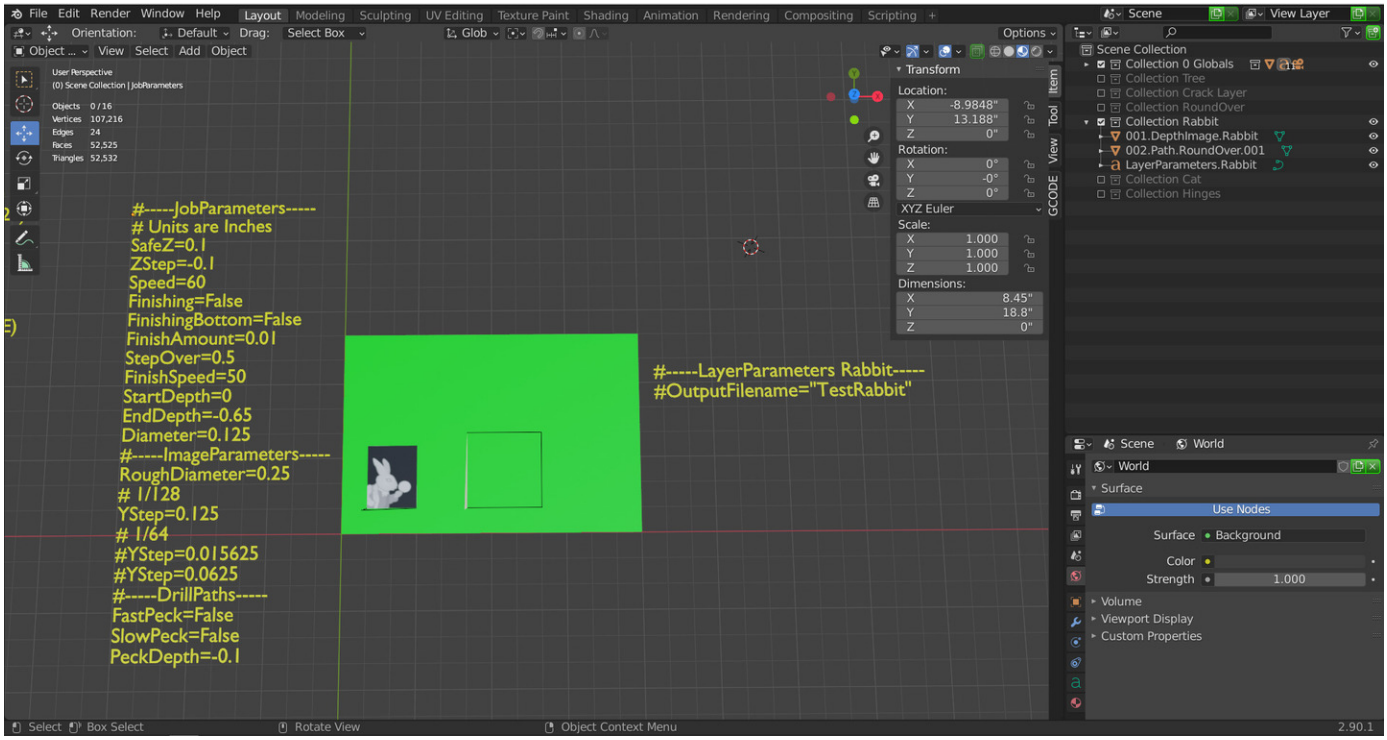

Figure 224: The collection "Collection Rabbit" carves a depth image of a rabbit and routes a rectangle path.

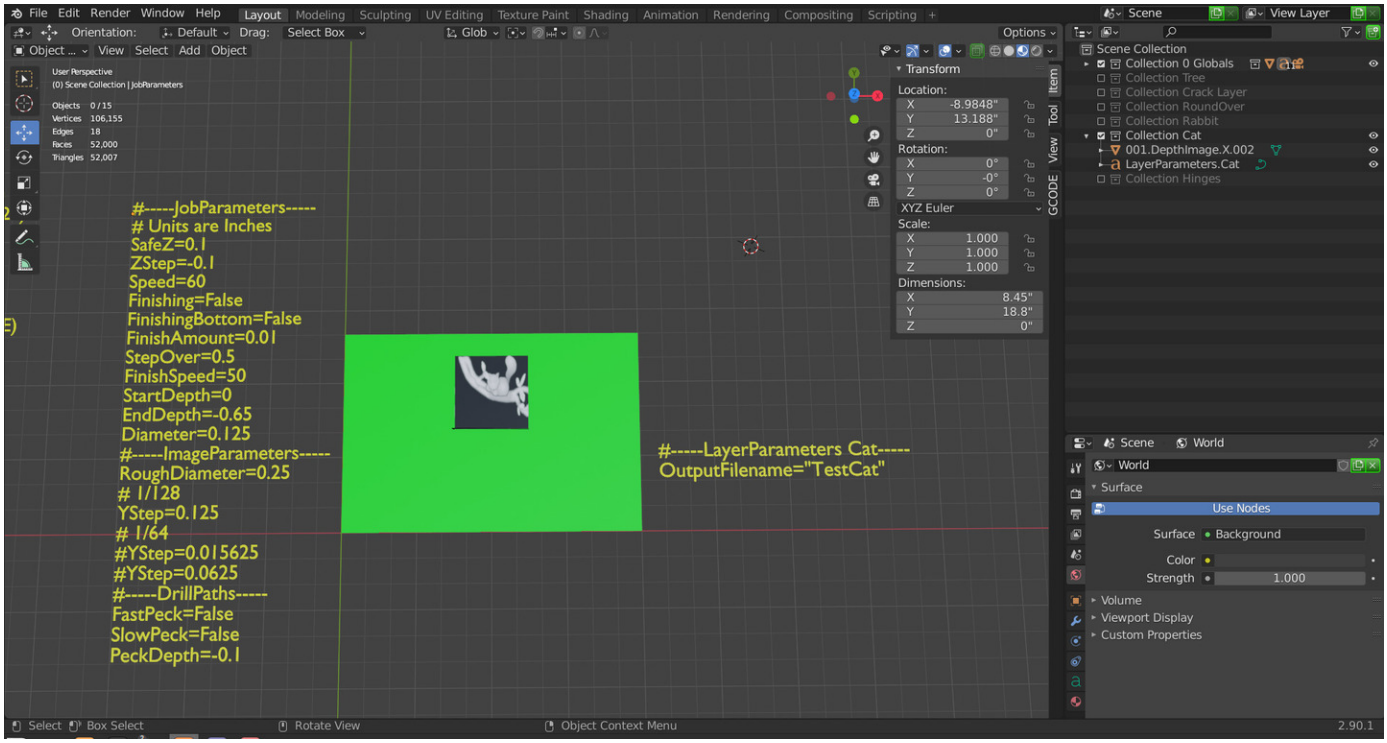

Figure 225: The collection "Collection Cat" carves a depth image of a cat.

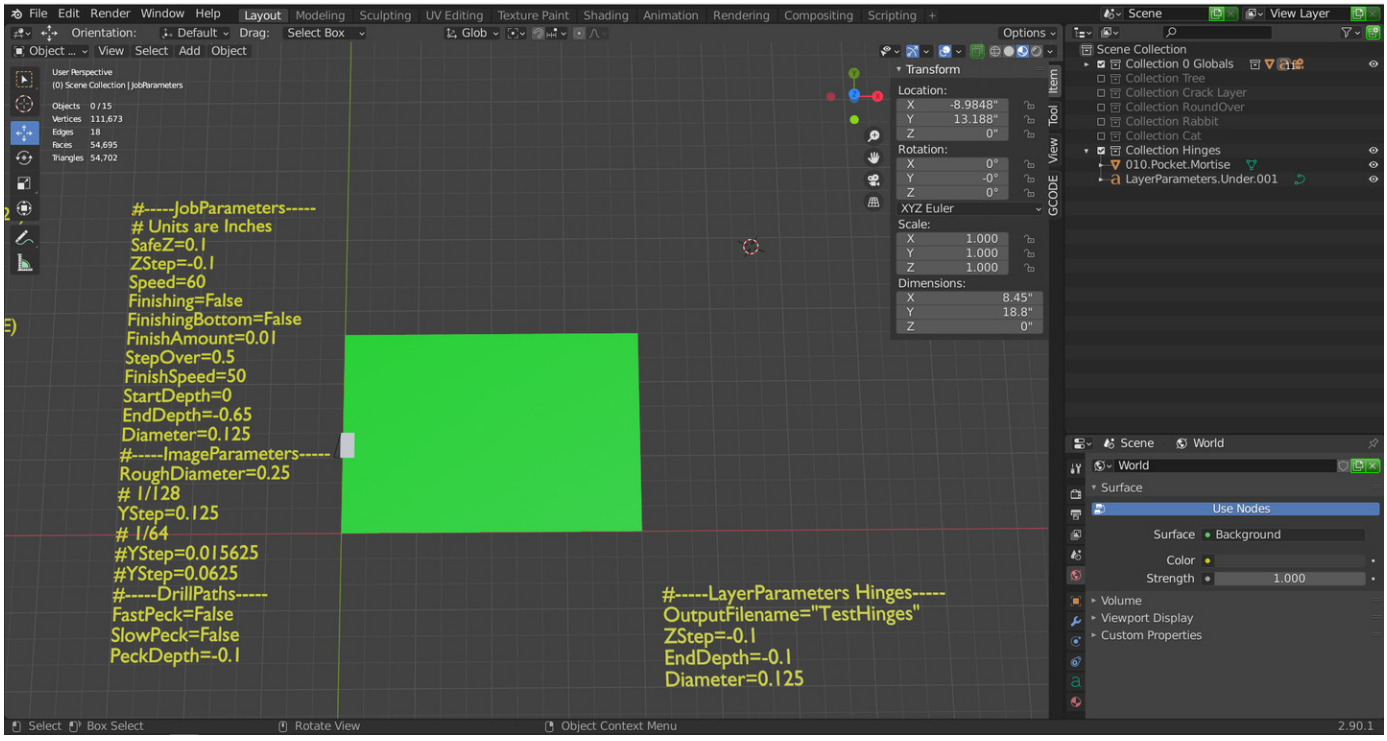

Figure 226: The collection "Collection Hinges" routes a small pocket.

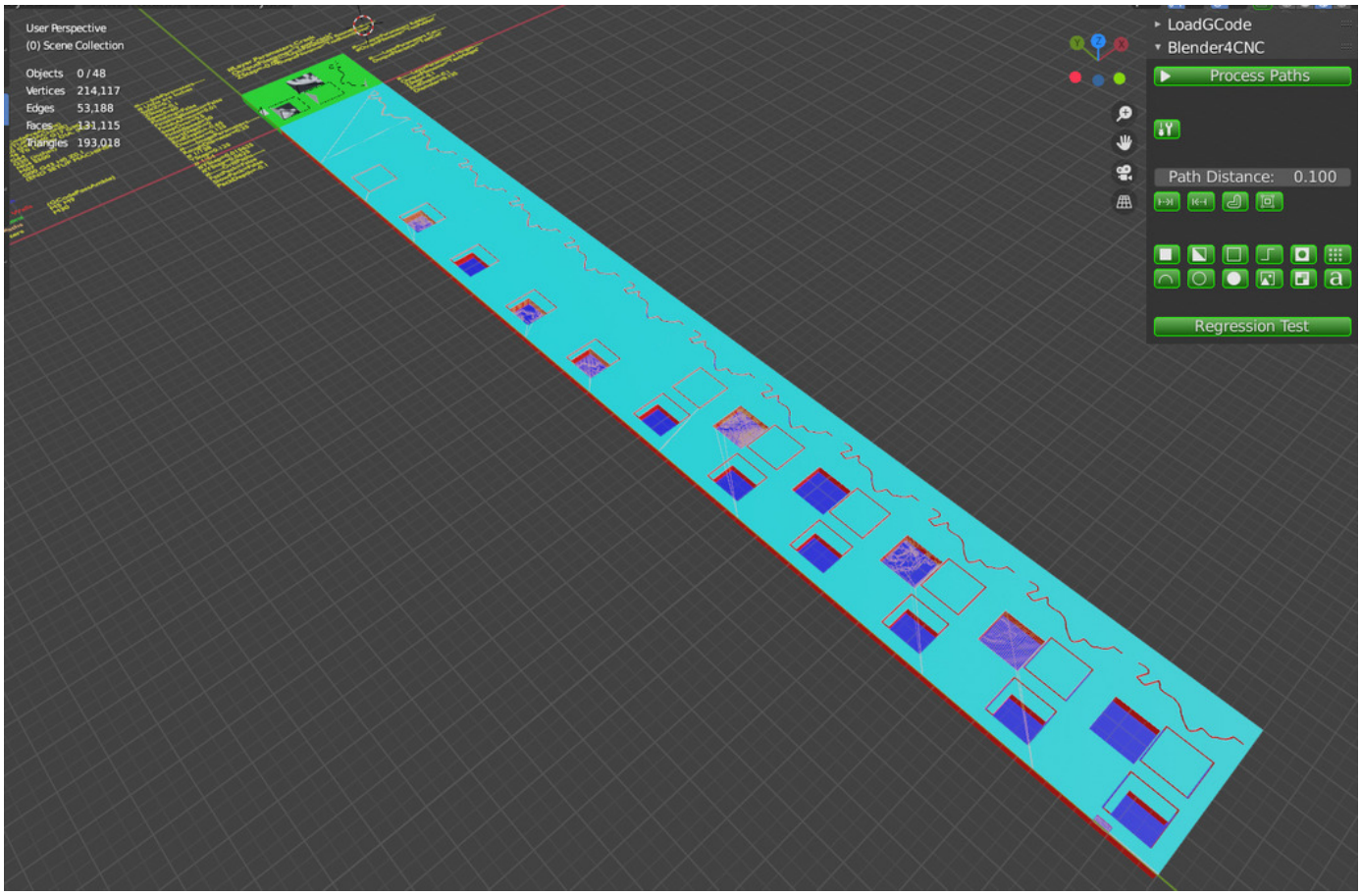

Figure 227: When all collections are enabled and the GCode is processed, we see all the steps laid out (made even longer because each depth image produces 4 stages).

Figure 228: Twelve GCode files are produced. (1) The "crack" path is cut with a 1/4" bit. (2) The "RoundOver" ractangle is cut with a 1/4" bit. (3) The 1st stage of cutting the rabbit depth image is done with a 1/2" bit; (4) the 2nd rabbit stage with a 1/4" bit; (5) the 3rd rabbit stage with a 1/4" round bit; and (6) the final rabbit stage is cut with a 1/4" round bit. (7) The rectangle in the "Collection Rabbit" is cut with a 1/4" flat bit. (8) The 1st stage of cutting the cat depth image is done with a 1/2" bit; (9) the 2nd cat stage with a 1/4" bit; (10) the 3rd cat stage with a 1/4" round bit; and (11) the final cat stage is cut with a 1/4" round bit. (12) The "Collection Hinges" pocket is cut with a 1/4" flat bit.

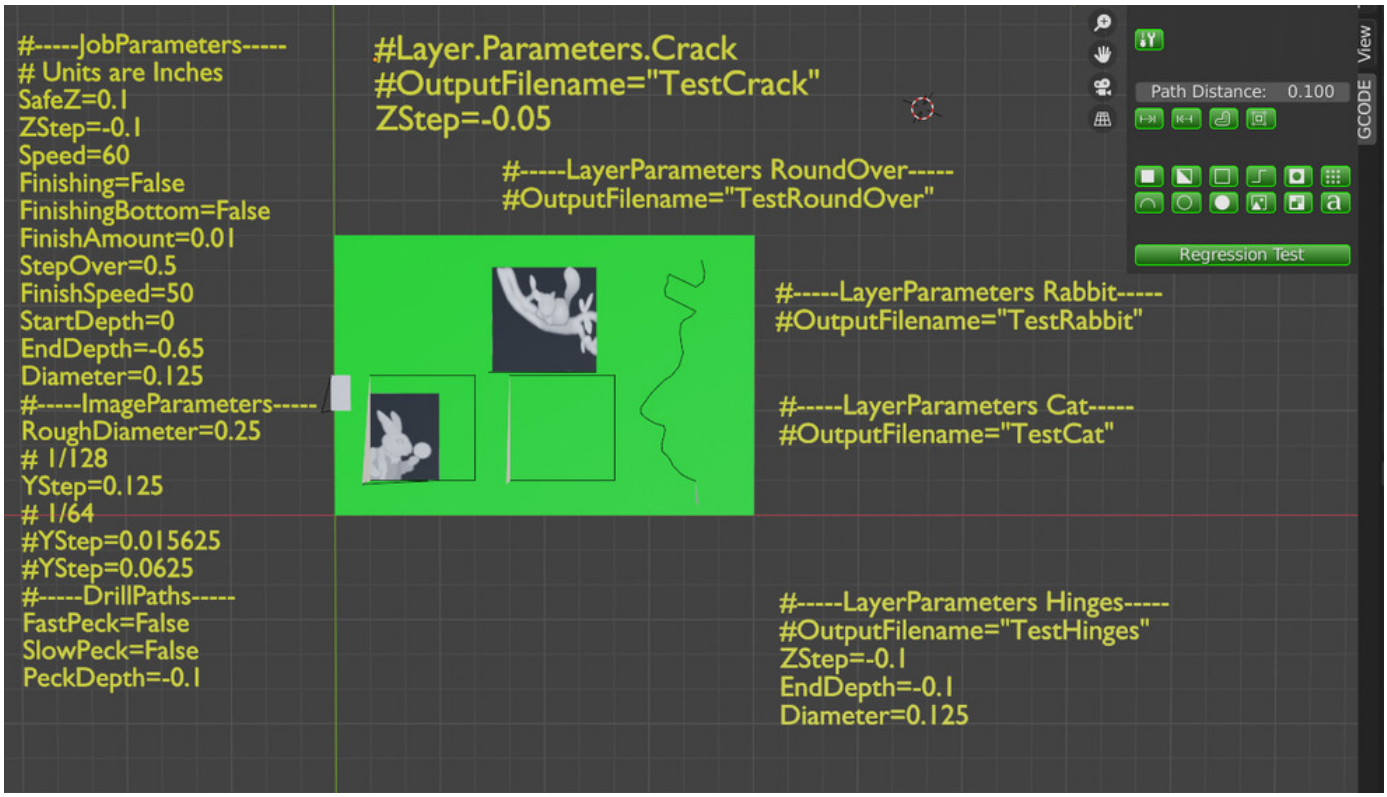

Figure 229: This time, we comment out all layer output filenames to see the difference in the output filenames produced for the GCode.

Figure 230: This time we only get 11 GCode files. Because none of the layer parameters define an output filename, the collection name is substituted. The original 1st and 2nd GCode files have been merged into a single GCode file because they both use the same diameter bit and the program chose to use the last collection "Collection RoundOver" in the filename. The program will always try to put multiple collections into the same GCode file. There are 3 things which trigger the program to produce a separate GCode file: (1) a change in cutter diameter; (2) a layer defines the "OutputFilename" variable in its parameters; and (3) a depth image which by definition goes through cutter diameter/shape changes.

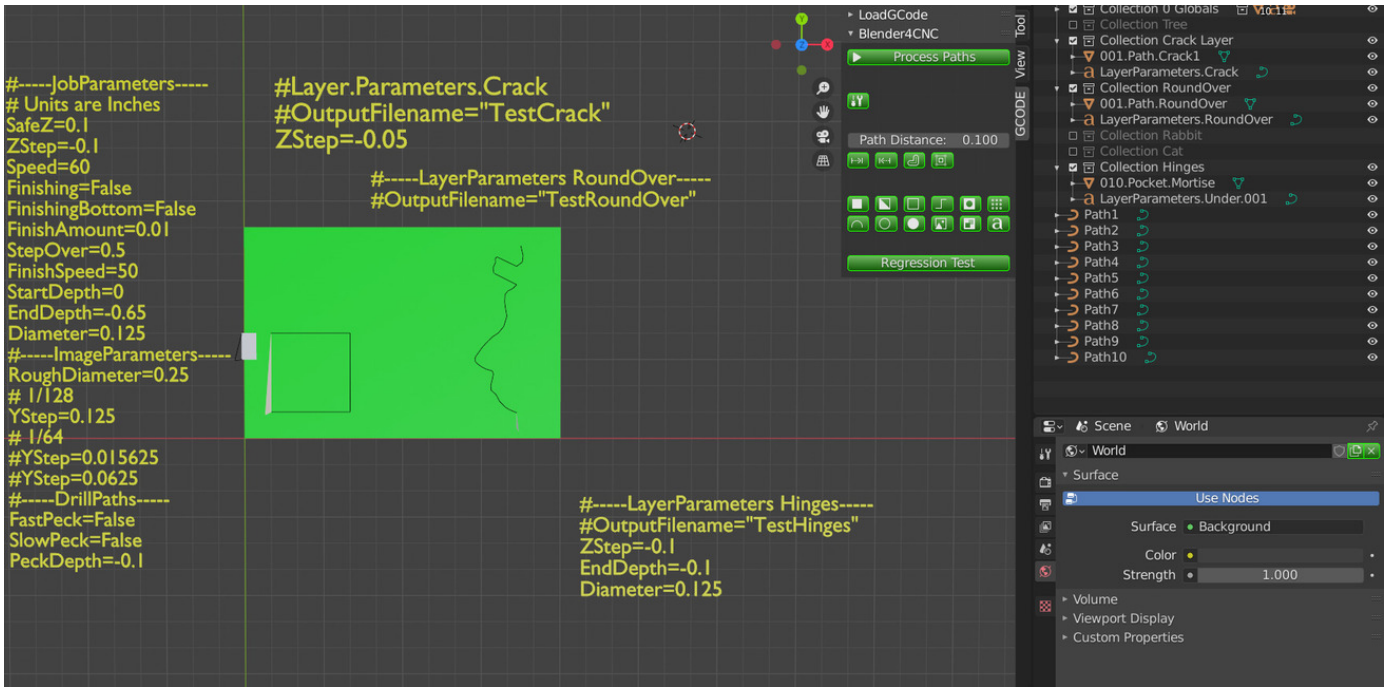

Figure 231: Here we only enable the three collections that contain no depth images and all use 1/4" diameter bit.

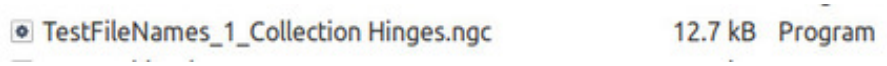

Figure 232: There is no need to change cutter bits between collections and operations therefore only one output file is produced.

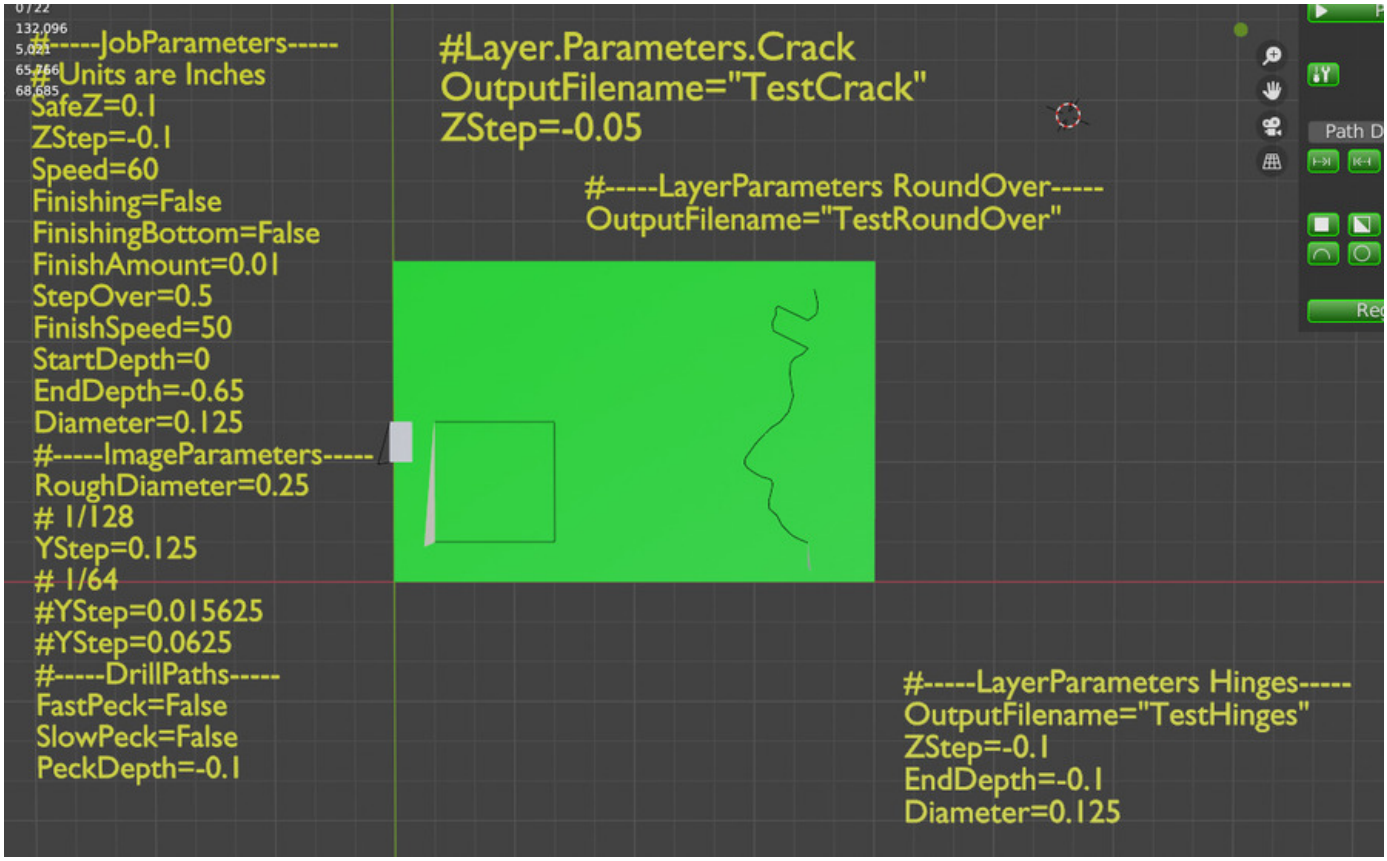

Figure 233: Let's uncomment the "OutputFilename" defined in each layer and produce GCode files.

TestFileNames_1_TestCrack.ngc	10.9 kB
TestFileNames_2_TestRoundOver.ngc	1.5 kB
TestFileNames_3_TestHinges.ngc	1.5 kB

Figure 234: Three output GCode files are produced (and named appropriately) because the defining of a name caused the program to create separate GCode files even though there were no change in bit diameter.

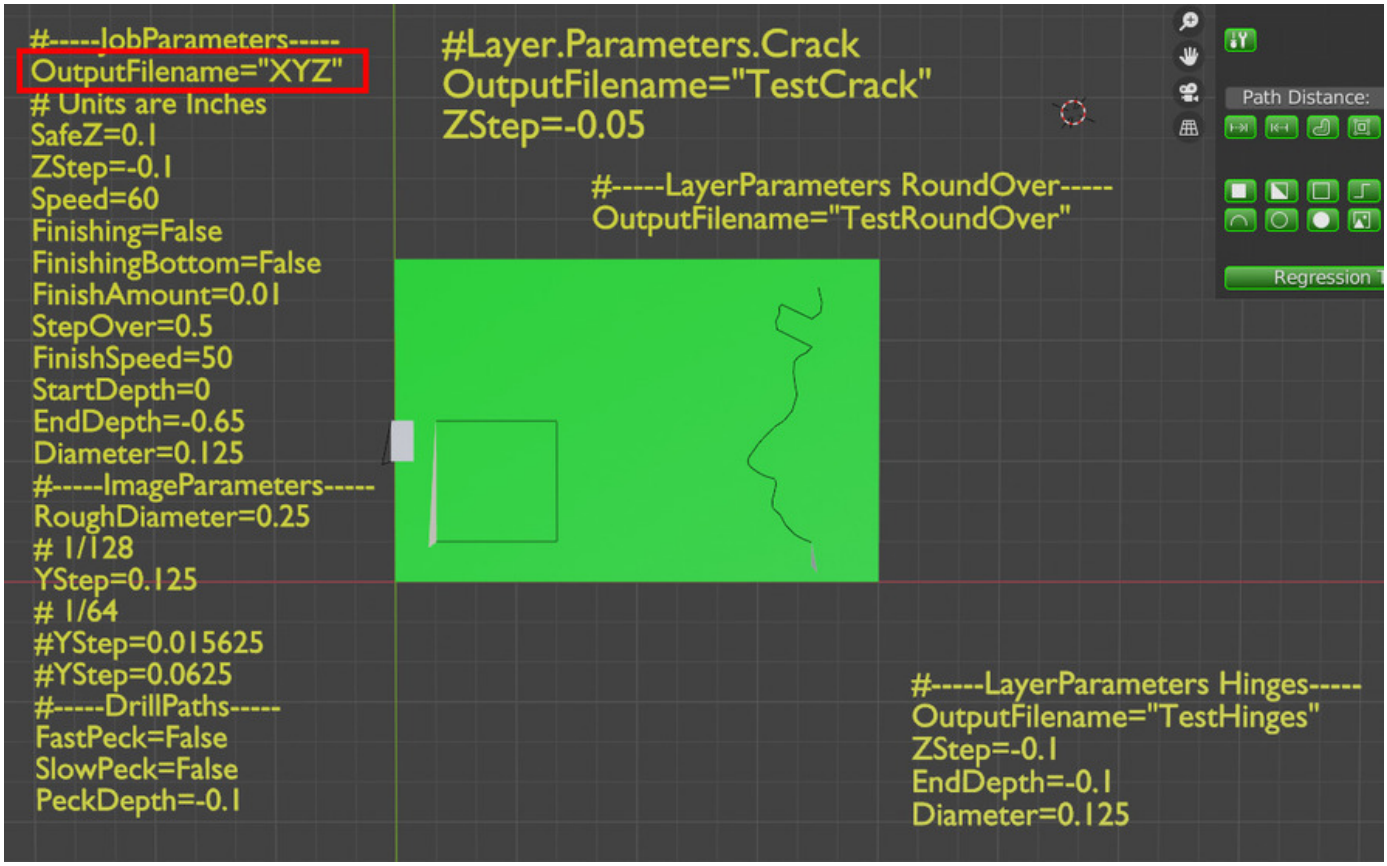

Figure 235: Notice that all the GCode files start with the name of the project "TestFileNames". This can be overridden by defining "OutputFilename" in the JobParameters.

Figure 236: Now all GCode files start with "XYZ" instead of "TestFileNames".

1.6.6 Project Parameters

The "Project Parameters" exists to handle things that are relevant to the type of project. It specifies a version so that any future versions of BLender4CNC can handle backwards compatibility should it be necessary. It also helps handle Blender attributes for the differences between metric and imperial projects by setting certain variables that help with visualizing CNC operations at typical scales in imperial or metric units.

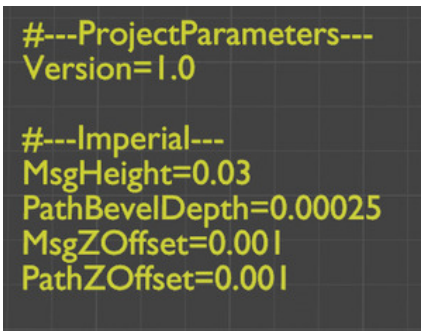

Figure 237: This is a set of Project Parameters for an imperial project. The version is set to 1.0 (the only option so far). "MsgHeight" is used by Blender4CNC to set the height of any error messages that need to be displayed. A value of 0.03 seems reasonable for working in the scale of inches. PathBevelDepth, MsgZOffset and PathZOffset are all used to help display router paths appropriately. You can change these values if you wish but be aware that things may look very strange if you do.

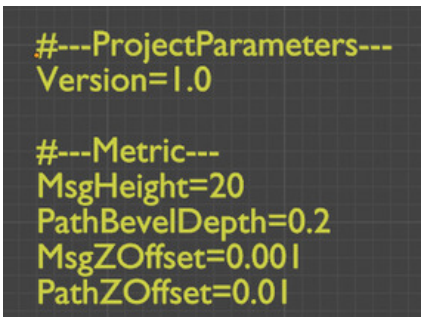

Figure 238: This is a set of Project Parameters for a metric project. Note how the values are different to the values for an imperial project - this is because the project is now working in meters (metres) or mm. You can change these values if you wish but be aware that things may look very strange if you do.

1.7 Blender4CNC Error Messages

Blender4CNC will try to produce helpful error messages when something goes wrong. It places a red error message close to the object that it **thinks** caused the error. If the error is more general it will appear at the origin (0,0).

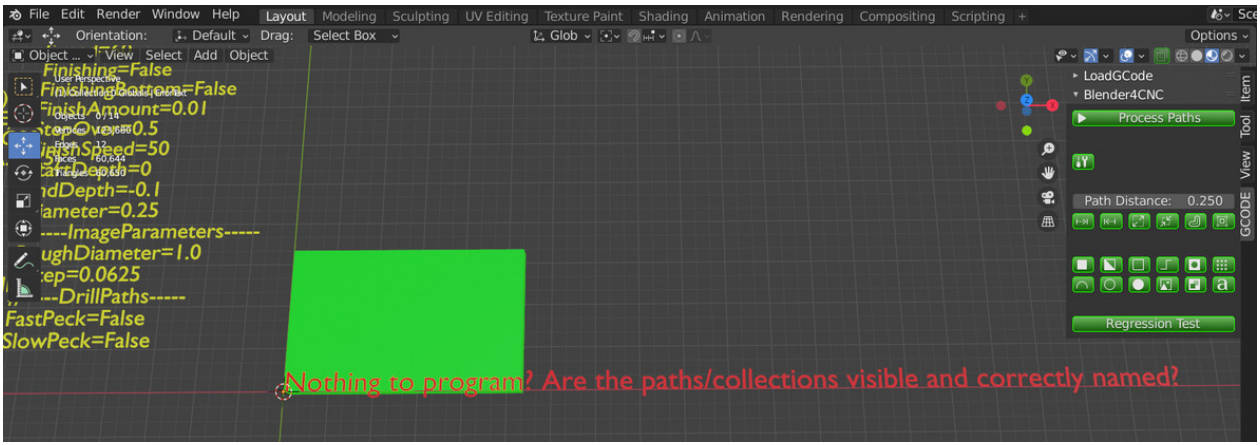

Figure 239: In this example, all operations were made "invisible" and then "Process Paths" was clicked. Since no operations are visible, there is nothing to do. The error message hints to the user that maybe they have accidently turned off all operations. Since this is a general error it appears at the origin.

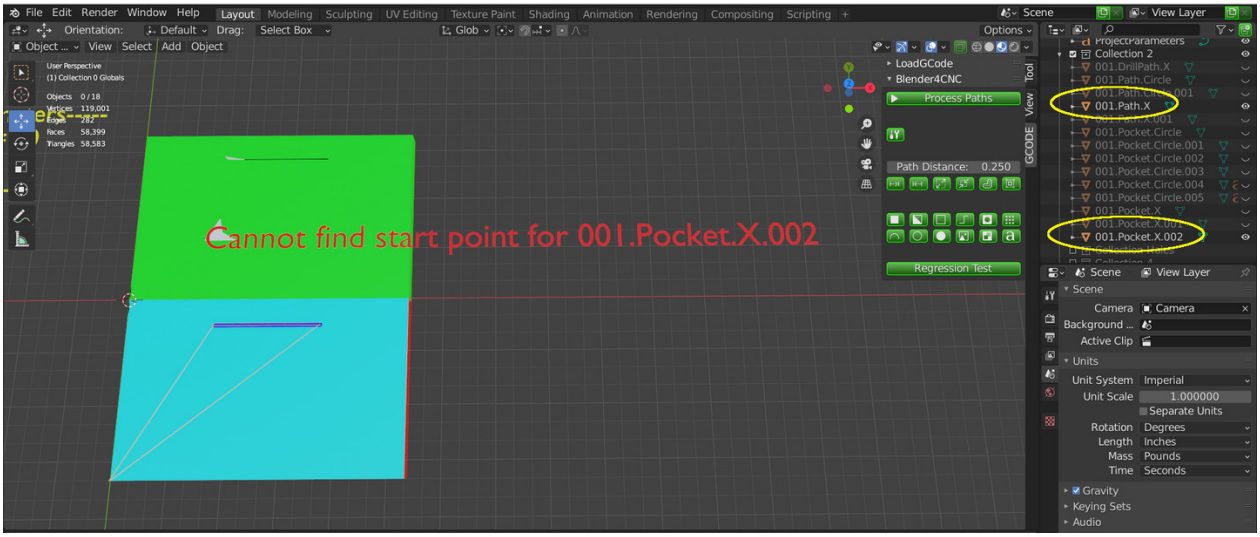

Figure 240: In this example, a pocket (called "001.Pocket.X.002") was created improperly without a "start" point. The error message has been placed at the origin of the erroneous operation and Blender4CNC has continued to process and visualize any remaining operations (such as "001.Path.X").

1.8 Working with Operations

1.8.1 Organizing Collections and Hiding Operations

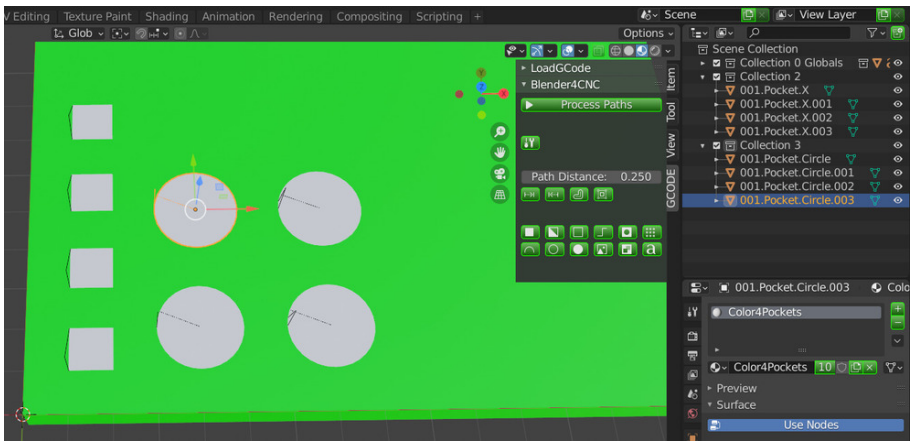

Figure 241: You can use blender collections to manage projects. Here is a project where all the rectangular pockets are in "Collection 2" and all the circular pockets are in "Collection 3".

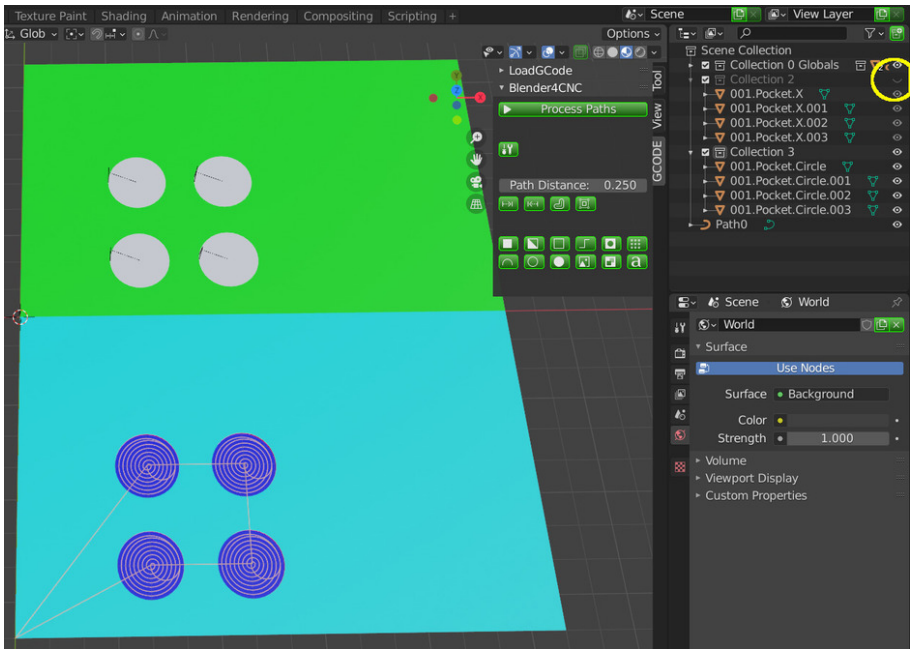

Figure 242: By clicking the "eye" icon beside a collection (far right of the screen), you can hide or show all theoperations in that collection. In this case "Collection 2" was hidden by clicking off the eye icon and when we click "Process Paths" only the operations under "Collection 3" are visualized. Only the operations for "Collection 3" are produced in the GCode.

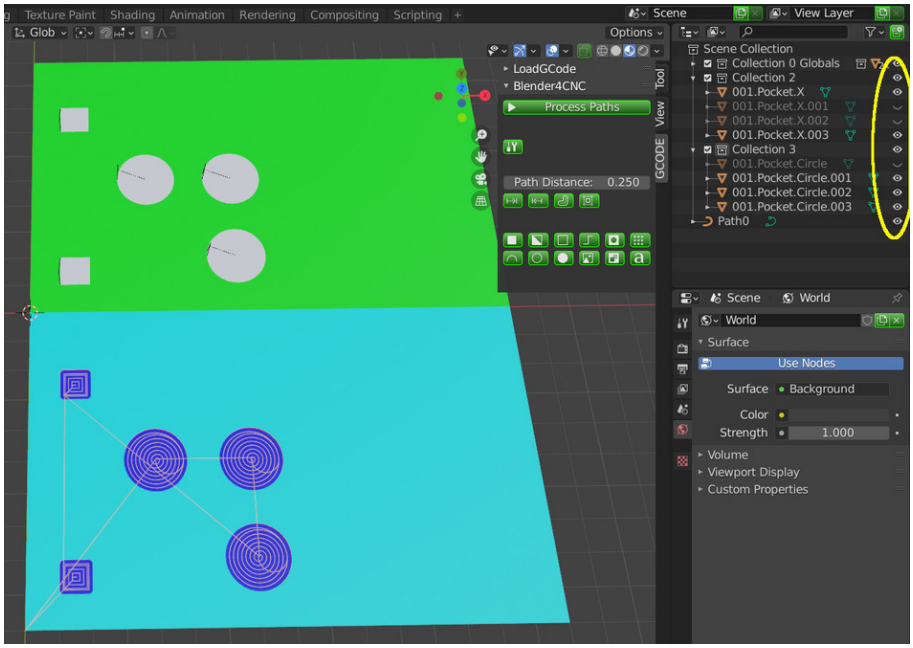

Figure 243: Likewise, you can click the eye icon beside any specific operation to enable/disable that operation. Here you can see that we disabled two of the square pocket operations and one of the circular operations by making them "invisible" with clicking the eye icon.

1.8.2 Copy Operations

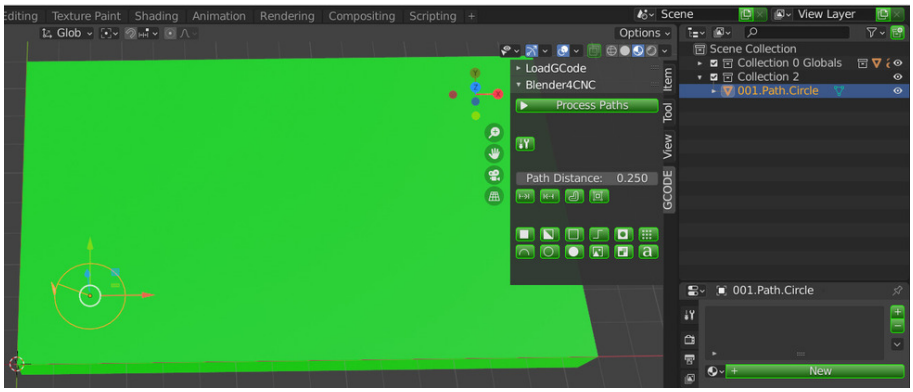

Figure 244: What if you have designed a shape and now you want to cut out many of the exact same shape in a sheet? Use the Blender "Duplicate" function. Here is an example of cutting many circles. Open up your template project file and add a circle path.

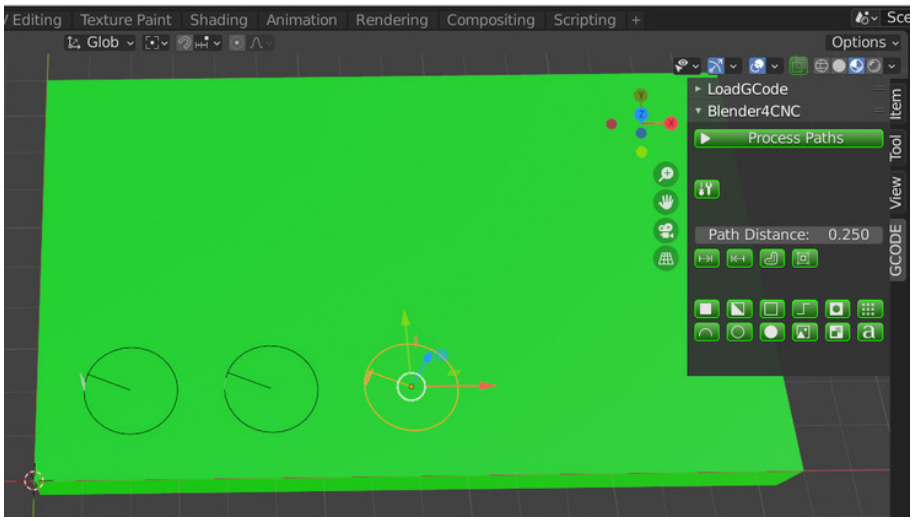

Figure 245: Press shift-d then move the copy to a different location. Then repeat once more to get 3 circles.

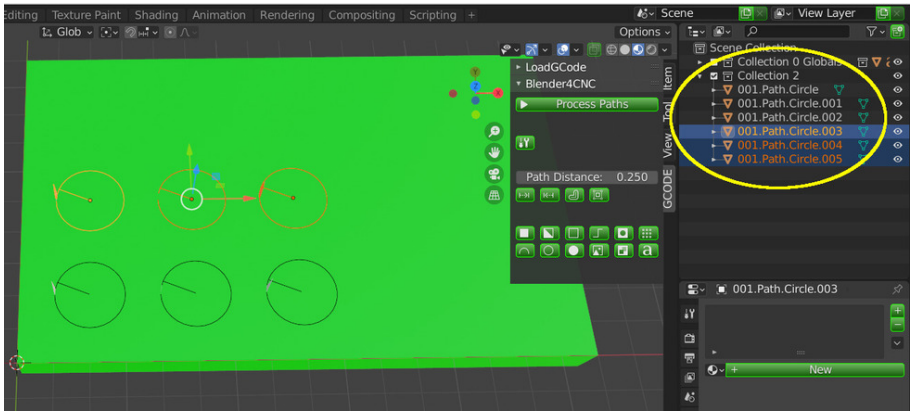

Figure 246: Now select all 3 circles (click the first circle then shift-click the other two circles) and press shift-d and move the three new copies. You will notice that each copy gets a number appended to the original name.

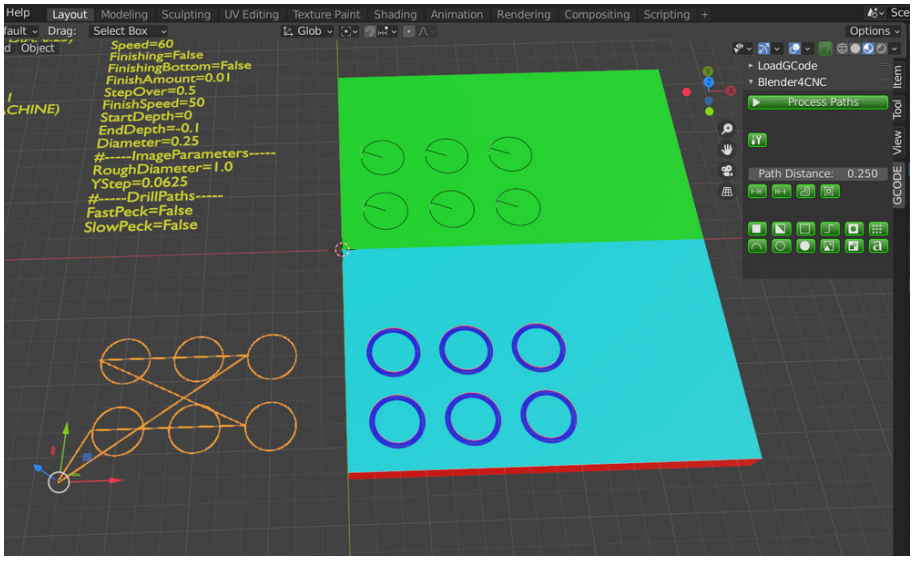

Figure 247: Multiple circles are cut.

1.8.3 Stretching Part of an Operation

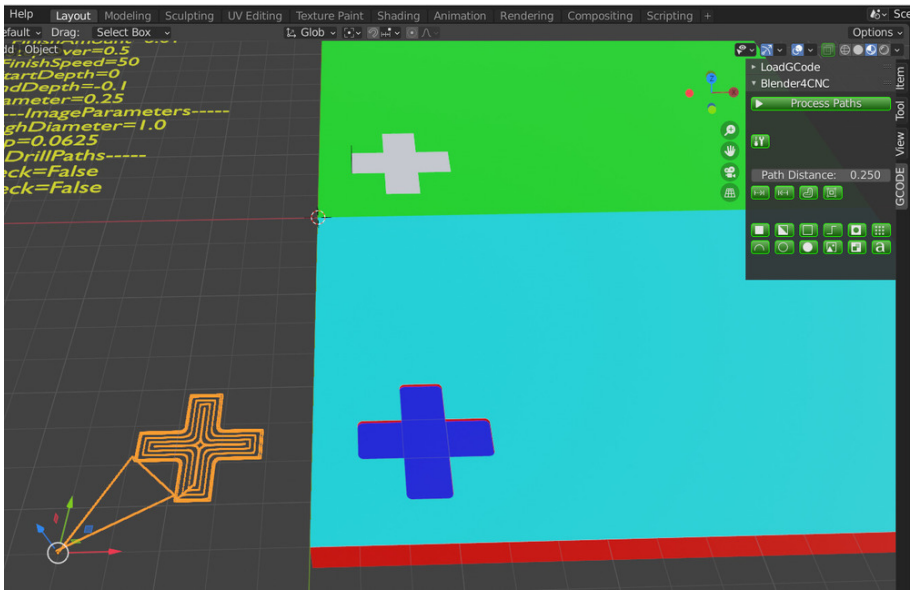

Figure 248: Let's say we want to extend the "right-hand" section of this cross shaped pocket.

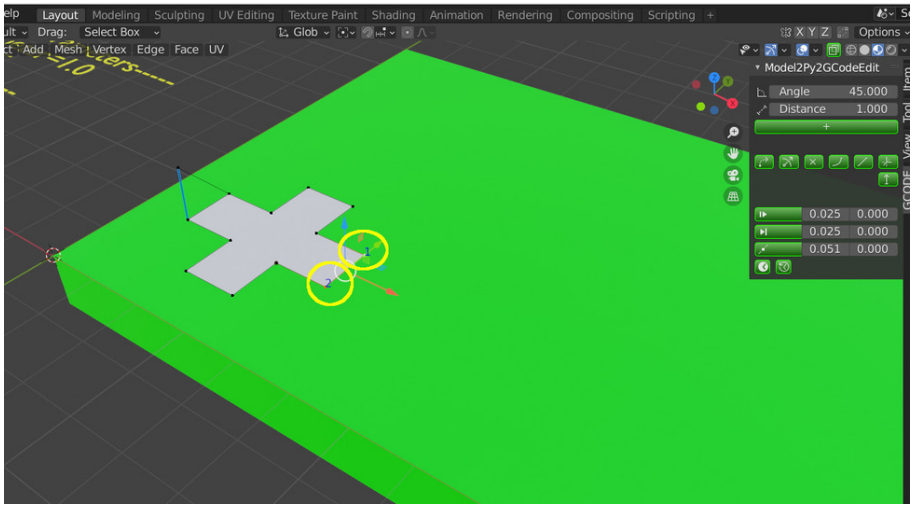

Figure 249: Select the pocket and enter edit mode (tab key) then select the two vertices.

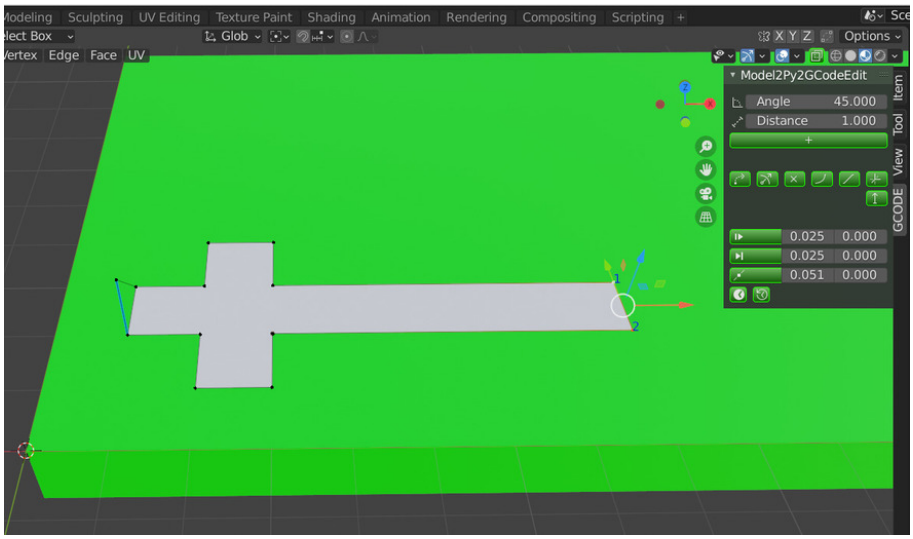

Figure 250: You can now grab the red arrow and drag the vertices further out.

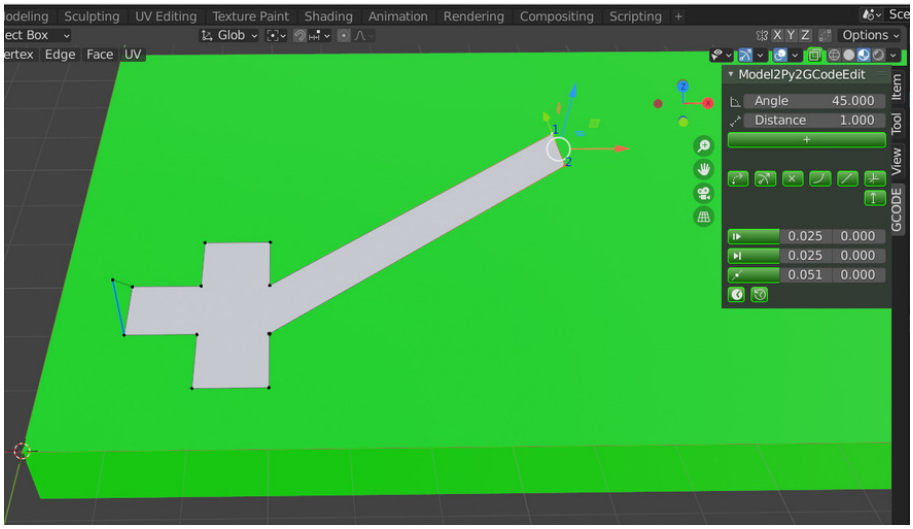

Figure 251: If you want, you can grab the green arrow and drag the vertices "up".

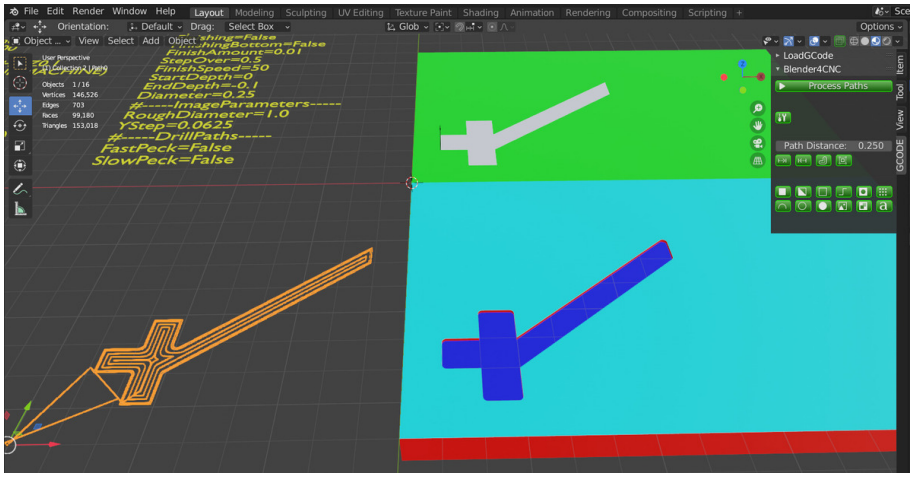

Figure 252: The new pocket shape

1.8.4 Resizing an Operation

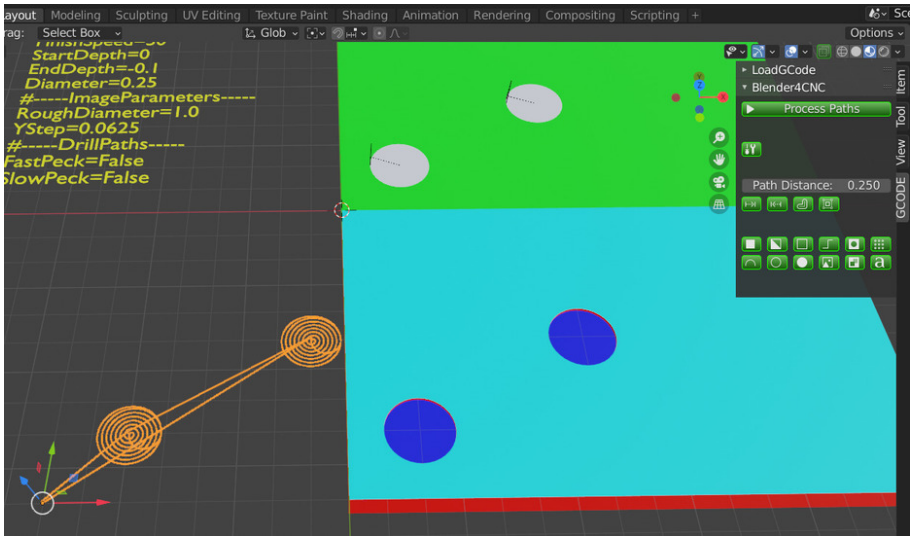

Figure 253: Let's make the circular pocket on the right larger.

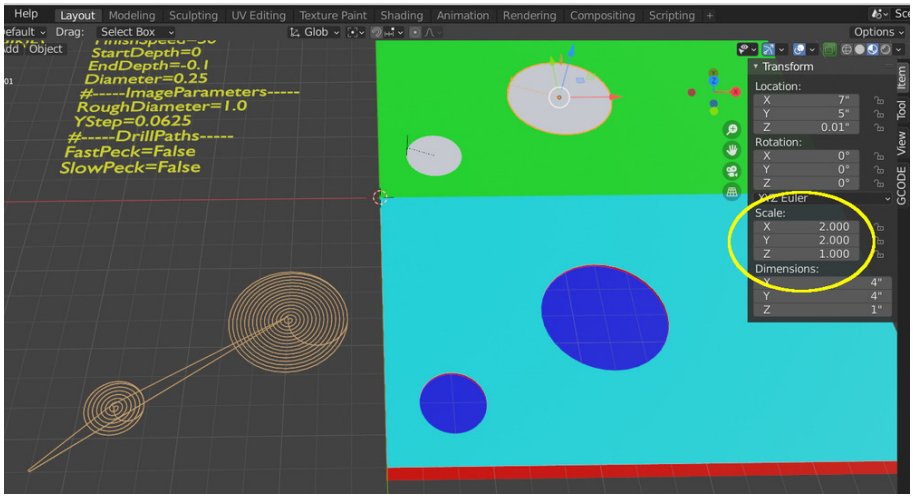

Figure 254: Select the pocket and change the X,Y dimensions.

1.8.5 Creating a path to the "left" or "right".

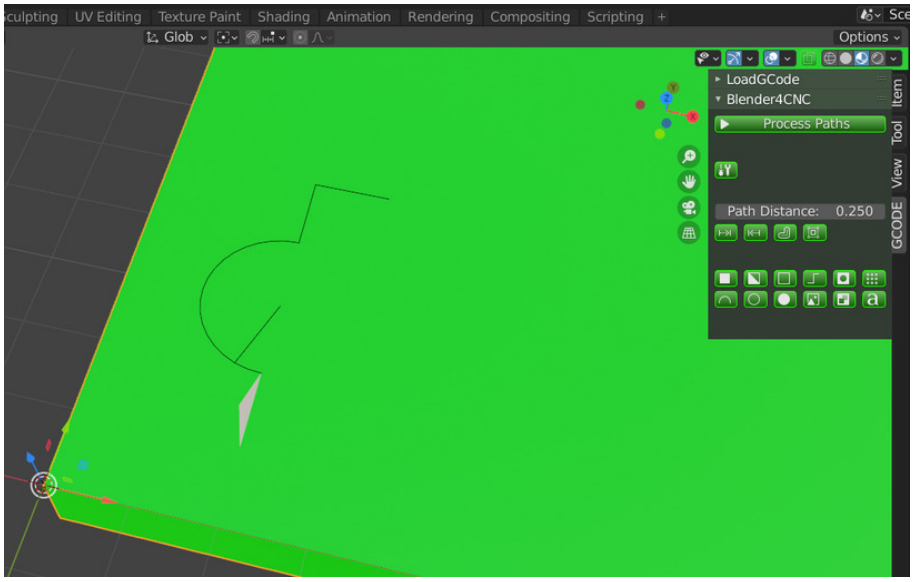

Figure 255: Sometimes you may wish to create a path that is exactly a certain distance away from another path. (Maybe you need to trace around an object at a small distance.) Here is an example path with curves and corners.

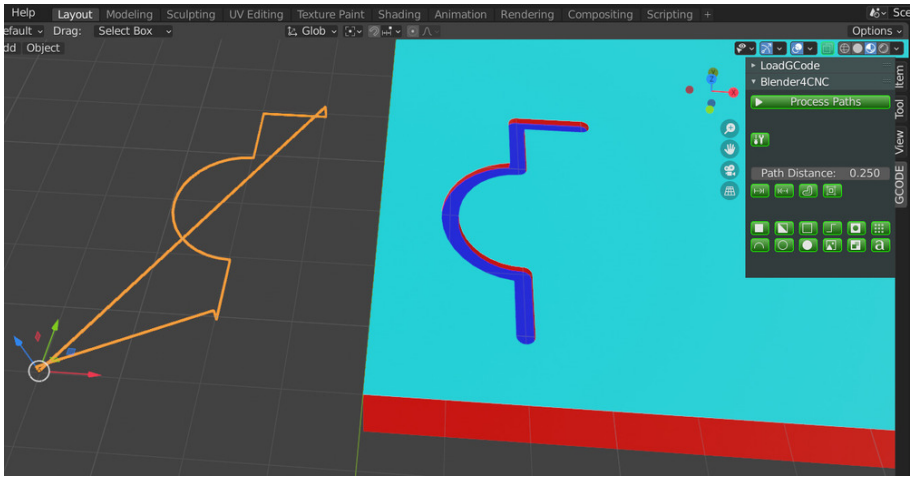

Figure 256: Here is how the path cuts the material.

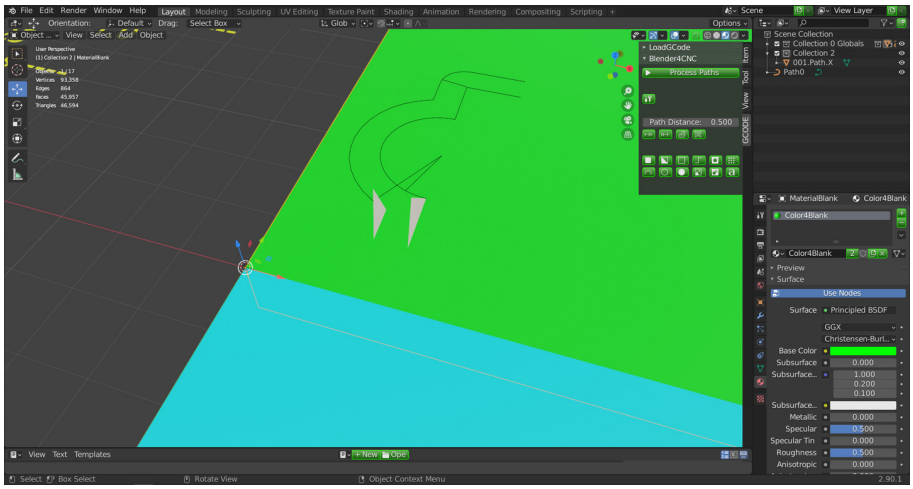

Figure 257: The path distance has been set to 0.5. Select the path and click on the "Create Path To Left" button. Notice that the new path is exactly 0.5 to the left of the original path and has generated curves when going around external corners.

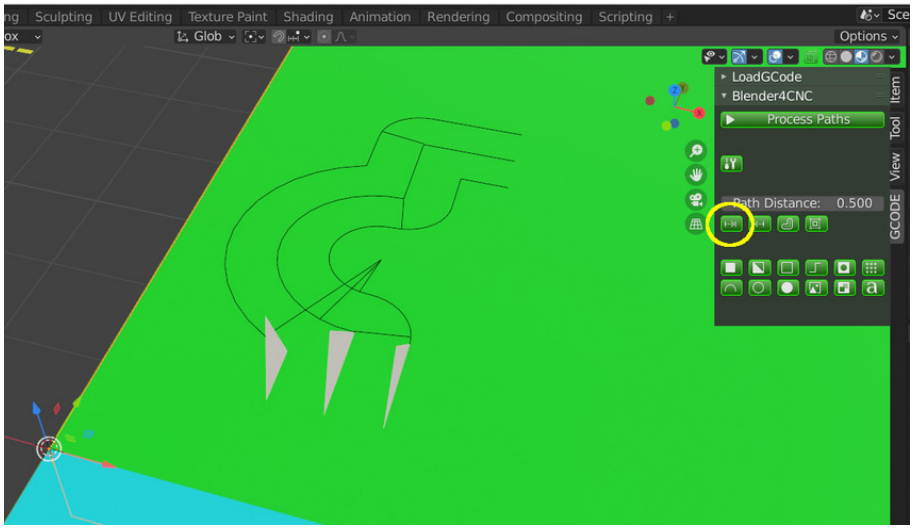

Figure 258: Select the original path and click on the "Create Path To Right" button. Notice that the new path is exactly 0.5 to the right of the original path. We now have three paths.

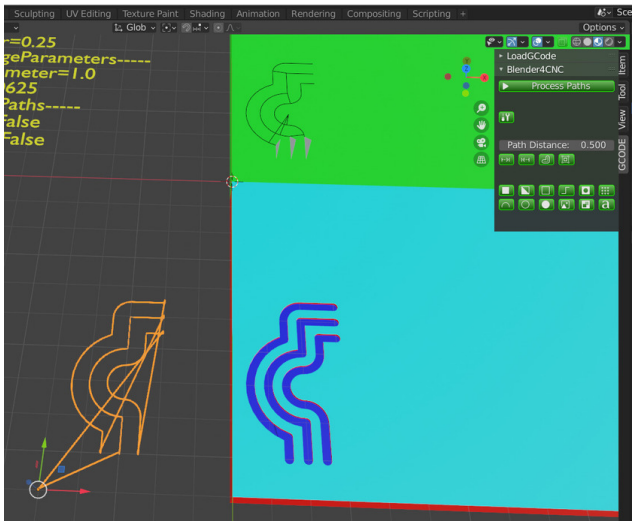

Figure 259: The three paths are visualized.

1.8.6 Setting the "origin" of an operation

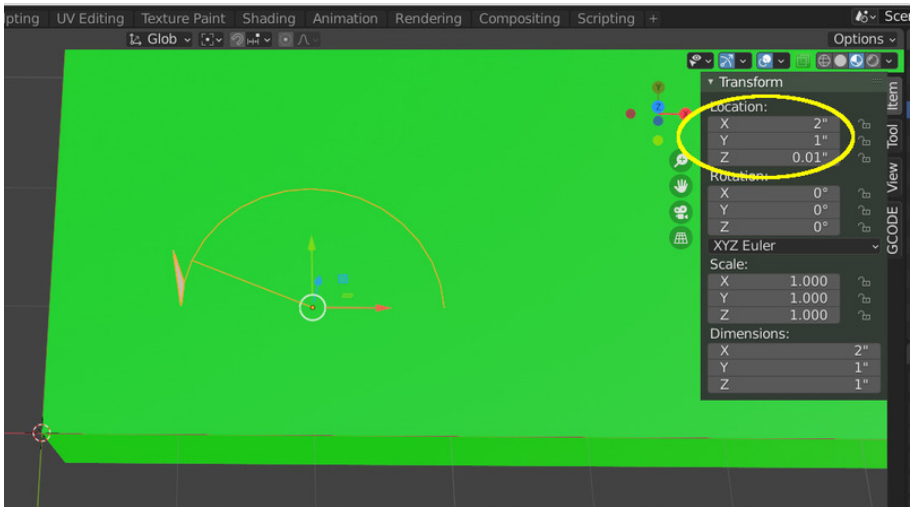

Figure 260: Each object in Blender has a "mesh" of vertices and that mesh has a (0,0,0) origin for itself (which may be a vertice or not). Most of the object generated by Blender4CNC default to defining the origin at the lower-left corner if rectangular, or the center of an arc. Sometimes you will want to change where the "origin" of an operation is for an object. Here is a curve. Notice how the location is set to X=2, Y=1 and this is where the center of the arc is put.

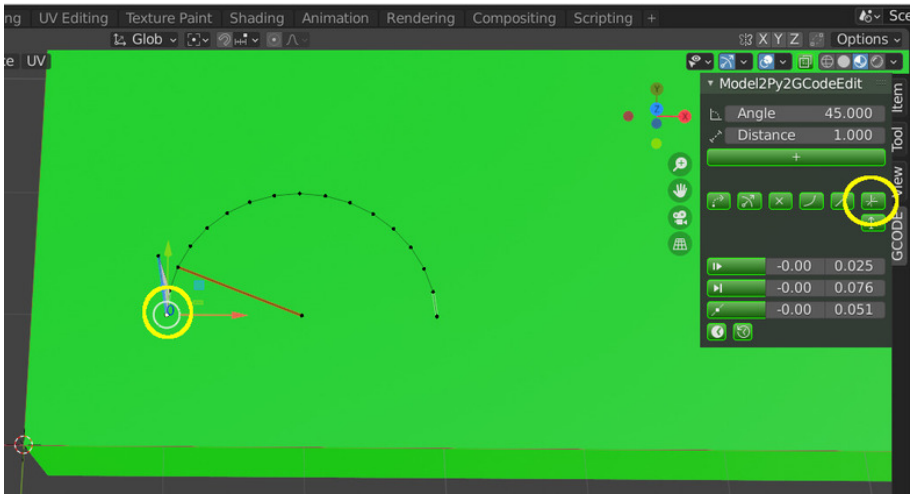

Figure 261: Let's say we want to set the origin of this curve to the starting point on the left. Select the curve and press TAB (edit mode). Then select just the first point and click the "Set Origin to Selected Vertex" button.

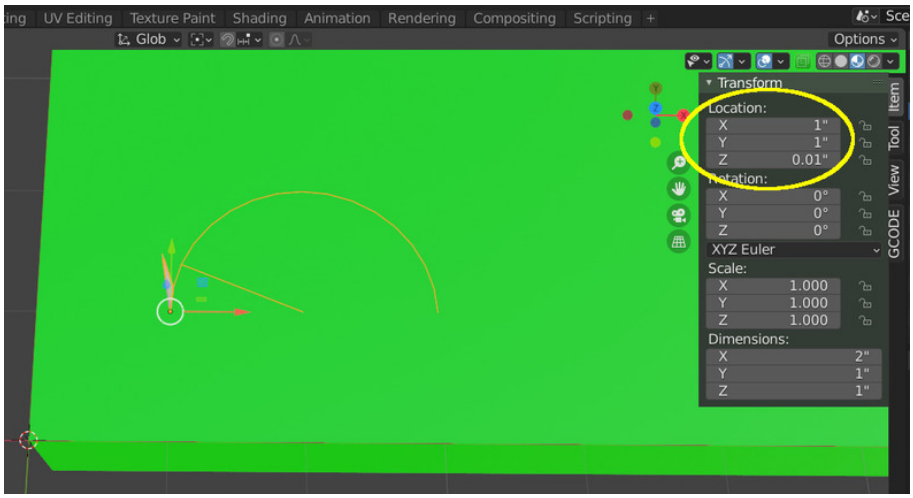

Figure 262: Press TAB to exit edit mode and notice now that while the arc has not moved, the location has changed to X=1 because the origin is now the left-most vertex.

1.8.7 Expand a Shape

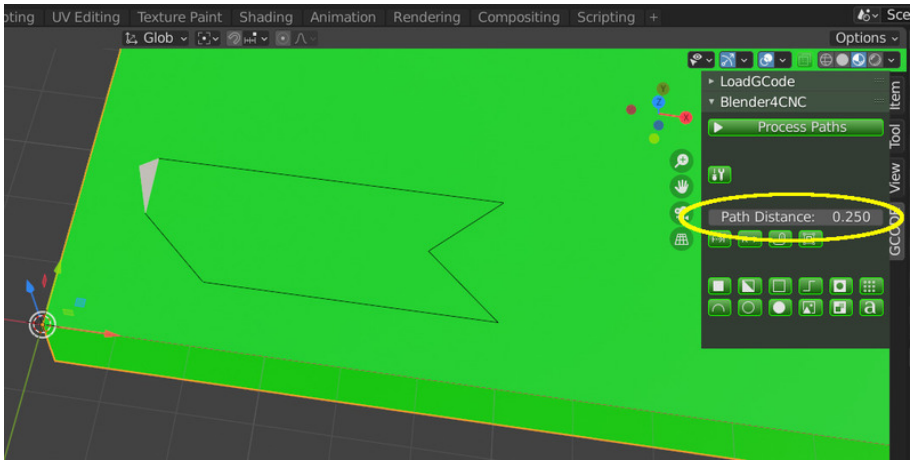

Figure 263: Sometimes you may want to expand a shape by a specific amount (maybe you want to trace around an object). Here is a shape that contains internal and external corners. The distance to expand by is set to 0.25. (The expand and shrink functions **only** work on pockets so make sure the shape has an appropriate name like "010.Pocket.A".)

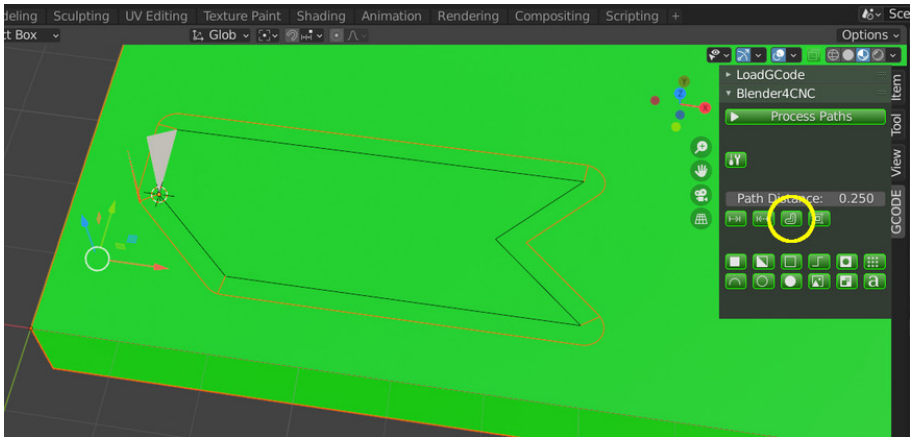

Figure 264: Click the "Expand Shape" button to create a 2nd path operation. Notice how it explicitly creates curves where the original shape had external corners. You can see that the "radius" lines of the curves all start from an external corner on the original shape. It creates a start point on the new path at a random location - you may have to move the start point if you desire it to be different.

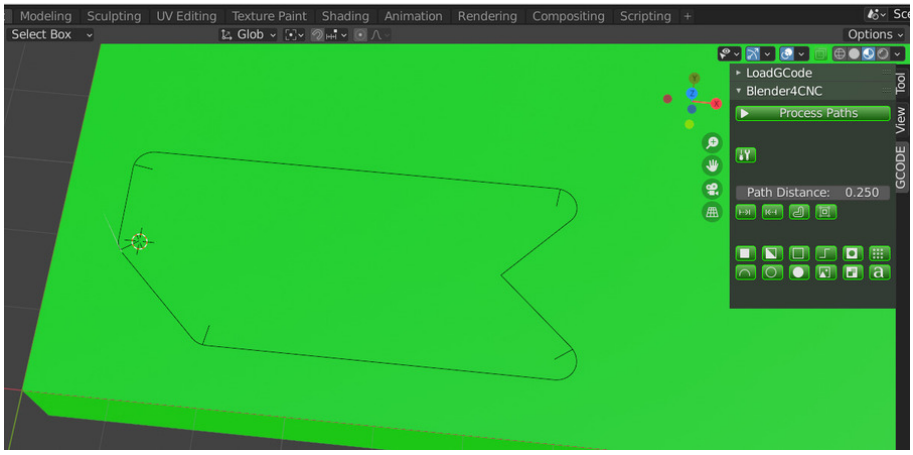

Figure 265: The original shape has been hidden to show the new path more clearly.

1.8.8 Shrink a Shape

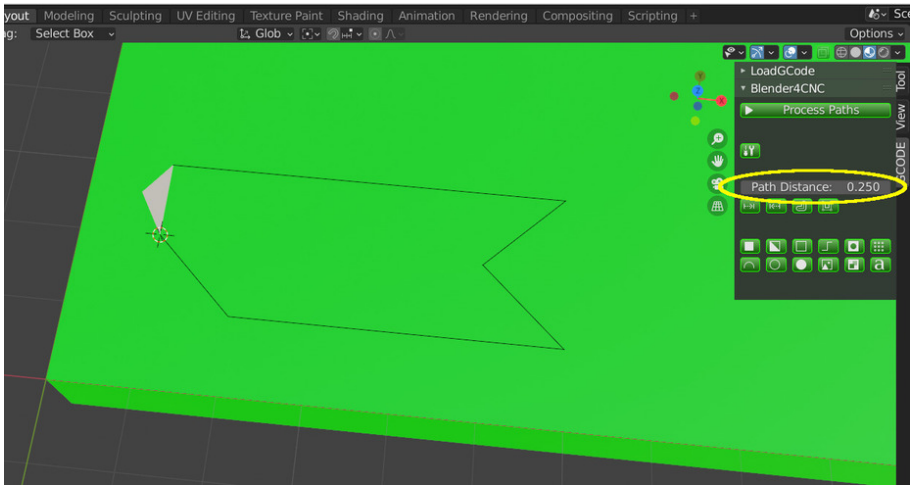

Figure 266: Sometimes you may want to shrink a shape by a specific amount. Here is a shape that contains internal and external corners. The distance to chrink by is set to 0.25. (The expand and shrink functions **only** work on pockets so make sure the shape has an appropriate name like "010.Pocket.A".)

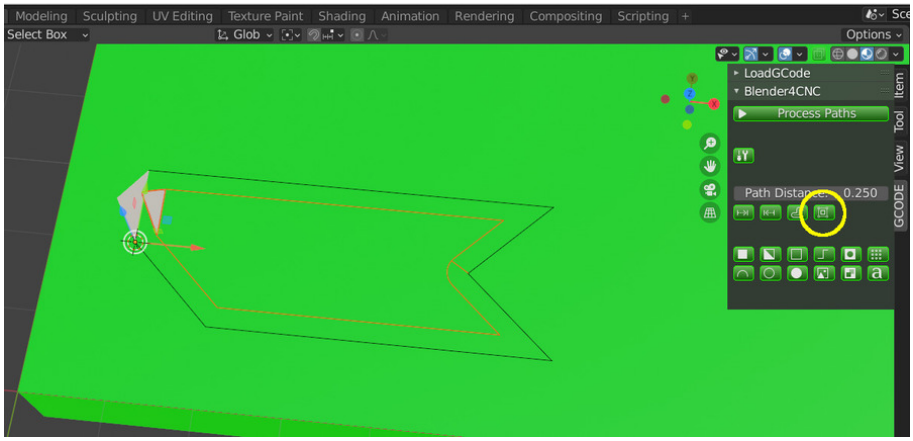

Figure 267: Click the "Shrink Shape" button to create a 2nd path operation. Notice how it explicitly creates curves where the original shape had internal corners. You can see that the "radius" lines of the curves all start from an internal corner on the original shape. It creates a start point on the new path at a random location - you may have to move the start point if you desire it to be different.

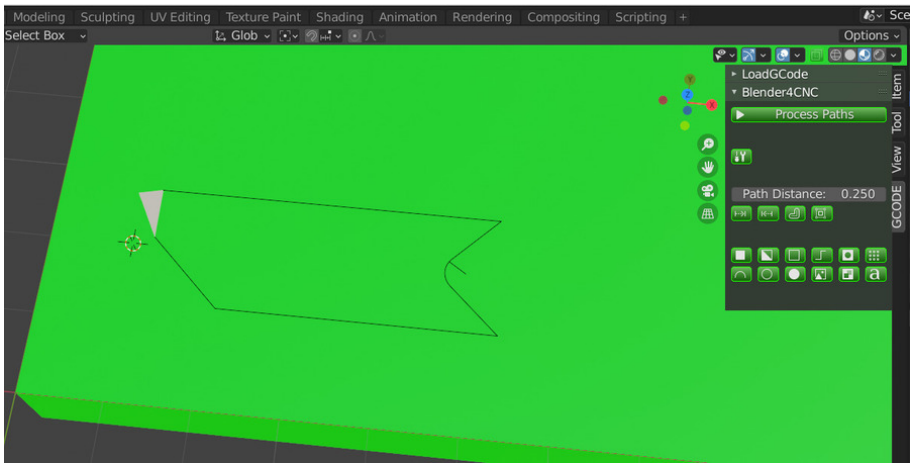

Figure 268: The original shape has been hidden to show the new path more clearly.

1.9 Working with Arcs

1.9.1 Marking the start/End of an arc

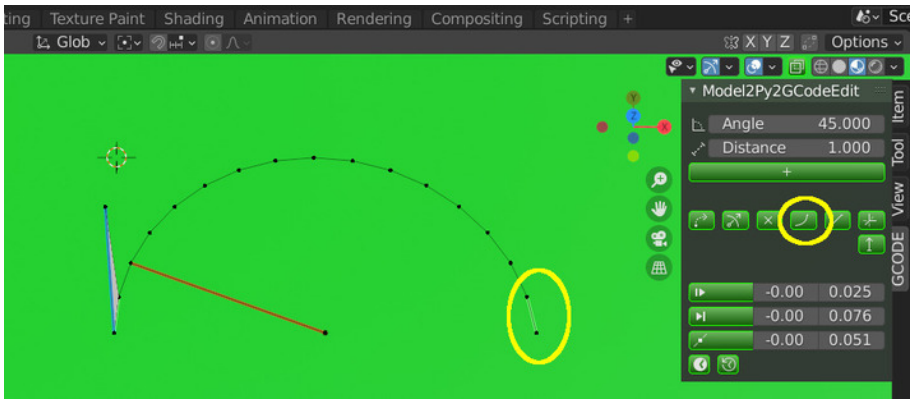

Figure 269: When in edit mode you can set a segment that marks the start or end of a curve by selecting the two points of that segment and clicking on the "Make Curve Segment" button (the segment turns green).

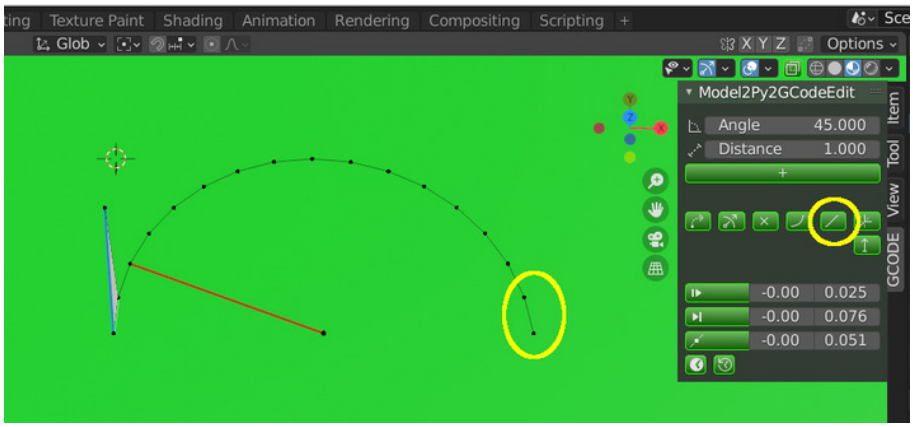

Figure 270: You can also "undo" a curve segment by selecting the two points of that segment and clicking on the "Unmake Curve Segment" button.

1.9.2 Marking a Radius

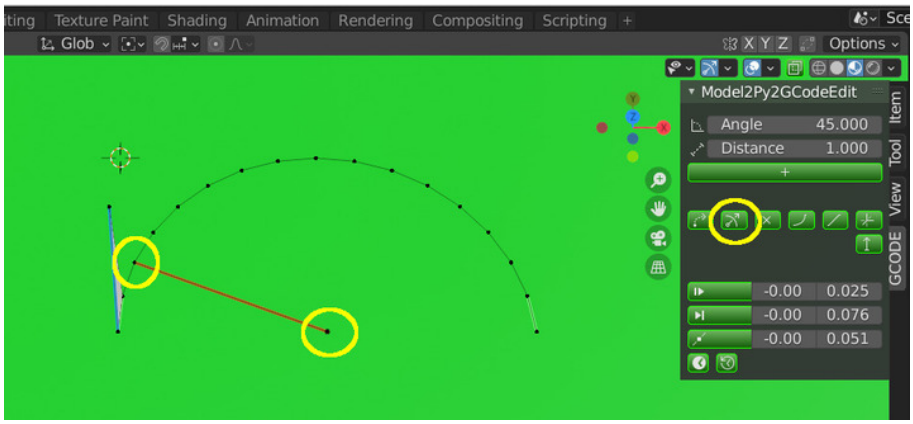

Figure 271: When in edit mode you can set a segment as a radius by selecting the two points of that segment and clicking on the "Make Radius Segment" button (the segment turns red).

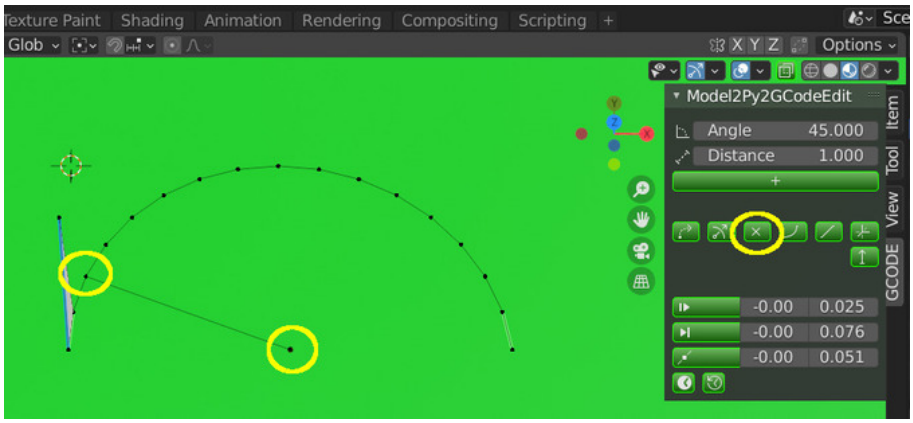

Figure 272: You can also "undo" a radius segment by selecting the two points of that segment and clicking on the "Unmake Radius Segment" button.

1.9.3 Creating an Arc from 3 points

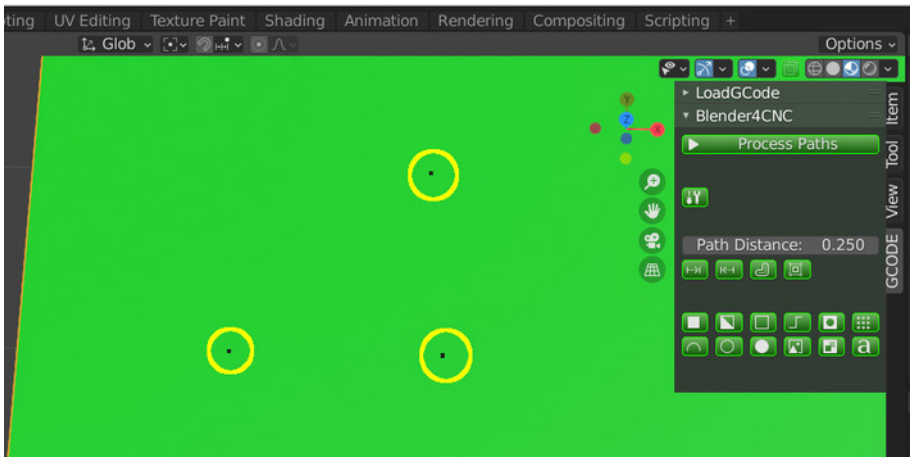

Figure 273: When in edit mode you can create an arc from 3 points - a start point, an end point, and the center of the arc. These points are at: start (1,1), end (2,2), and center (2,1).

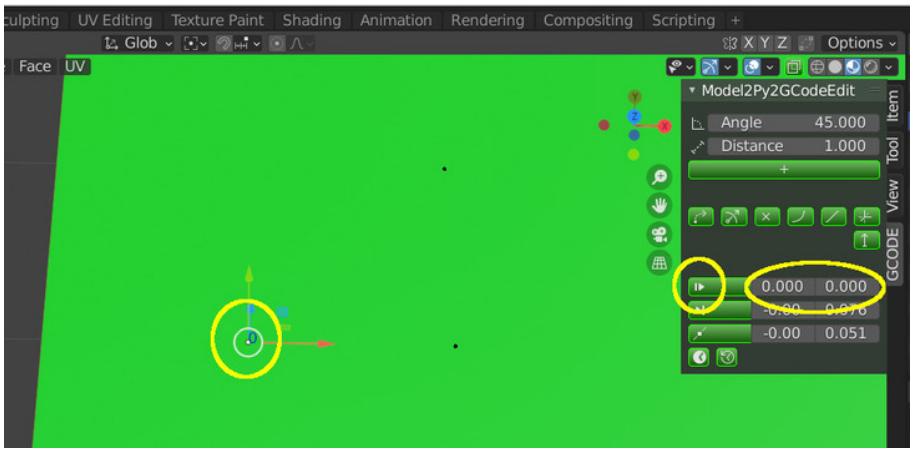

Figure 274: Select the start point of the arc and click the "Set Vertex as Start of Arc" button to specify the coordinates of the start point. The coordinates are shown as 0,0 because these are RELATIVE to the origin of the mesh. Since the origin of this mesh is the lower left vertex, the start point is the origin at 0,0.

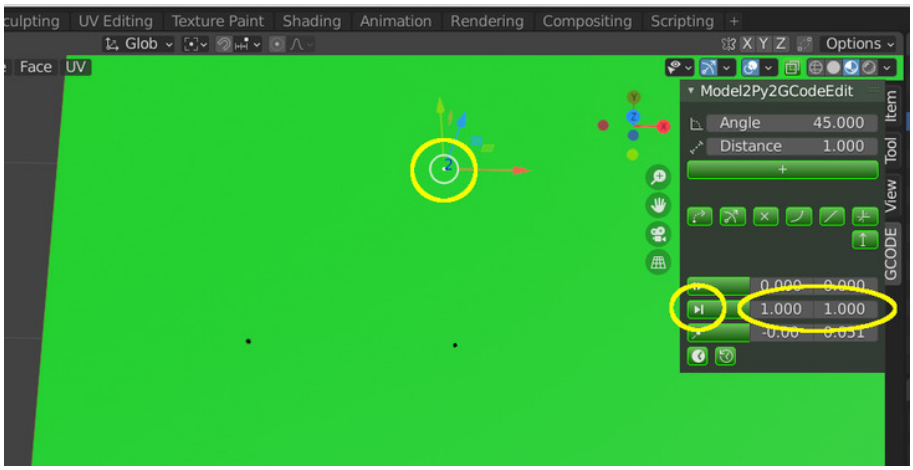

Figure 275: Select the end point of the arc and click the "Set Vertex as End of Arc" button to specify the coordinates of the end point. Again, the coordinates appear as 1,1 because these are relative to the origin.

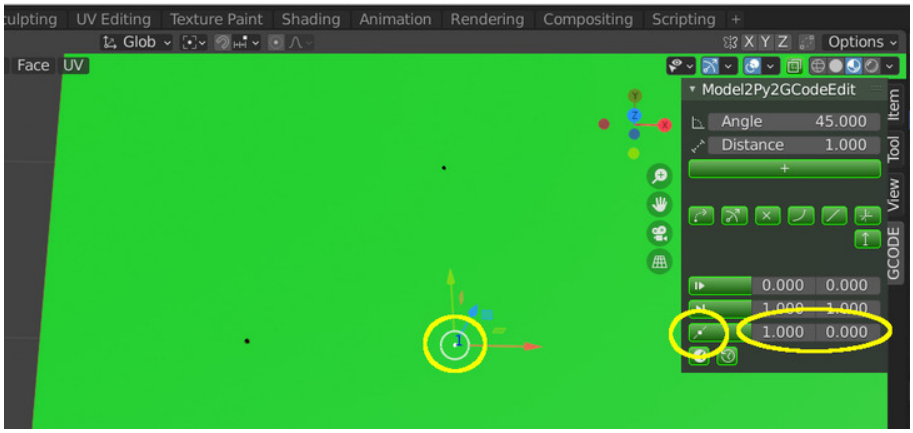

Figure 276: Select the center point of the arc and click the "Set Vertex as Center of Arc" button to specify the coordinates of the center point. Again, the coordinates appear as 1,0 because these are relative to the origin.

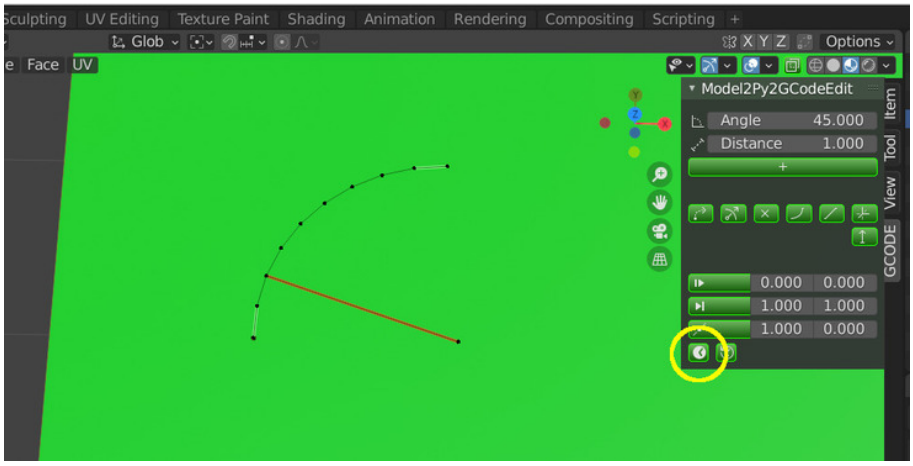

Figure 277: Click the "Create CW Curve" button to make a clockwise arc. Notice that a start point (with a blue segment) is not created because it is assumed that this curve may be in the middle of a path.

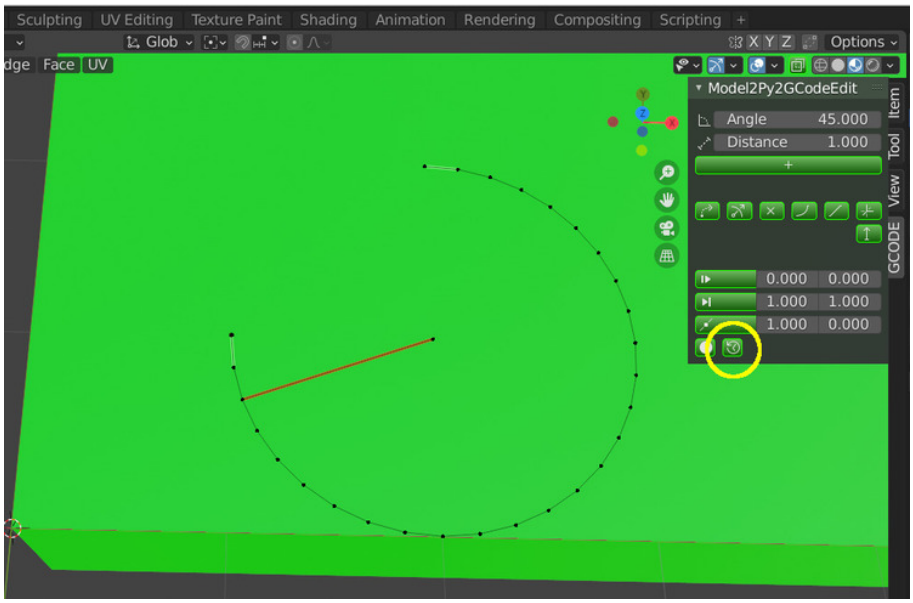

Figure 278: Click the "Create CCW Curve" button to make a counter-clockwise arc.

1.9.4 A note on the middle points of arcs

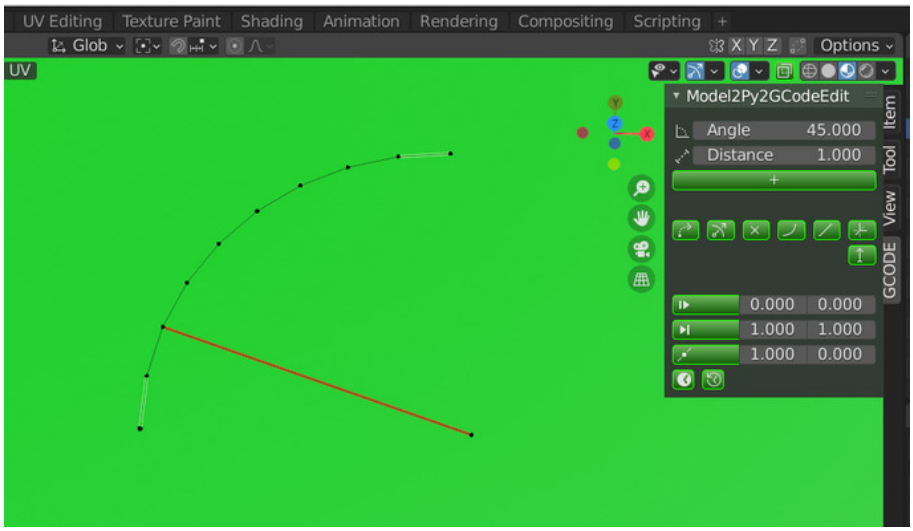

Figure 279: The "middle" points on a curve do not have to be exact. Here is a quarter circle arc.

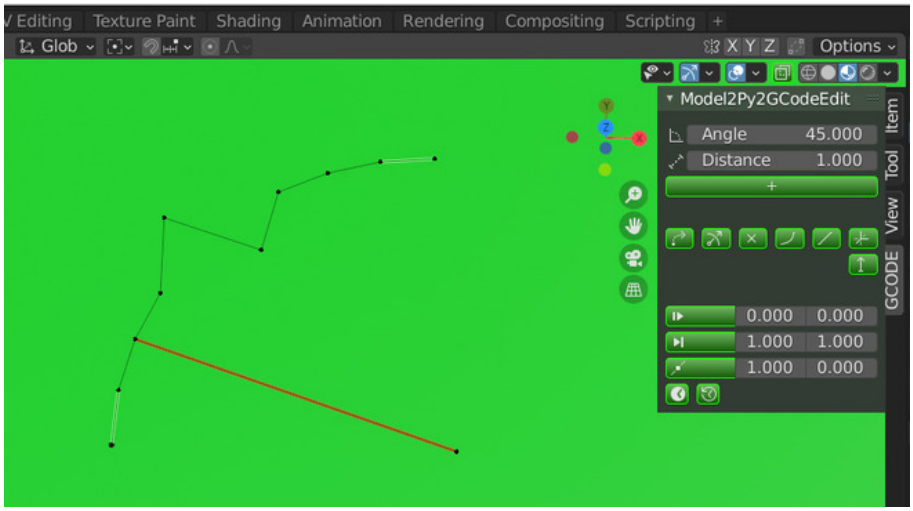

Figure 280: Some of the middle points have been dragged to a position which doesn't look nice (to over-exagerate). The curve is still defined by the start and end points and the radius etc.

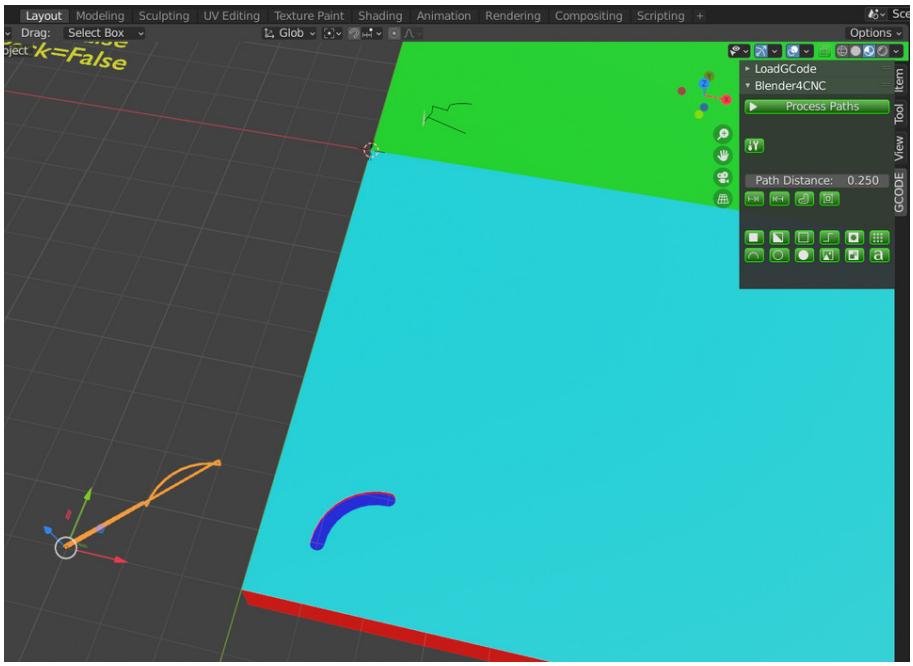

Figure 281: When visualized, the curve looks correct (and the generated GCode is a proper curve).

1.10 Working with Meshes for Paths and Pockets

1.10.1 Marking a Start Point

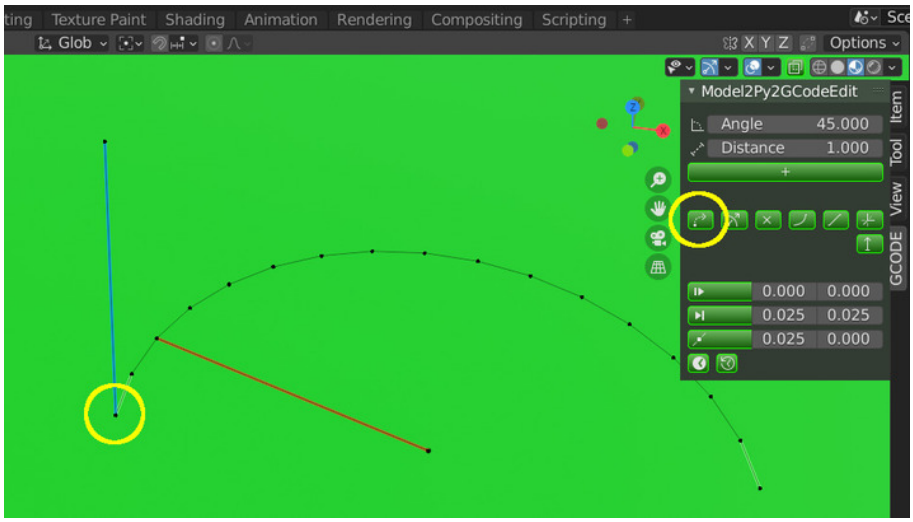

Figure 282: When in edit mode you can specify the start of a path or pocket by selecting the first vertex and then clicking on the "Make a StartPoint" button (a new vertical blue segment is added to the start point). To "undo" this action, select the vertex at the top of the blue segment and press the "Delete" key.

1.10.2 Creating a solid filled "Face"

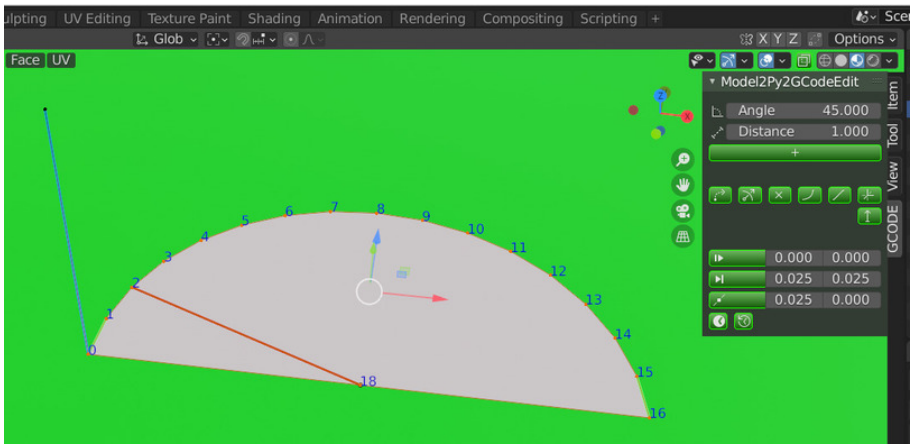

Figure 283: When in edit mode you can create a "face" by selecting a set of co-planar points (they must all lie on the same plane) and press "f". In this example, all the points were selected EXCEPT the vertex at the top of the blue segment. (More useful to indicate pockets than paths with a filled face.)

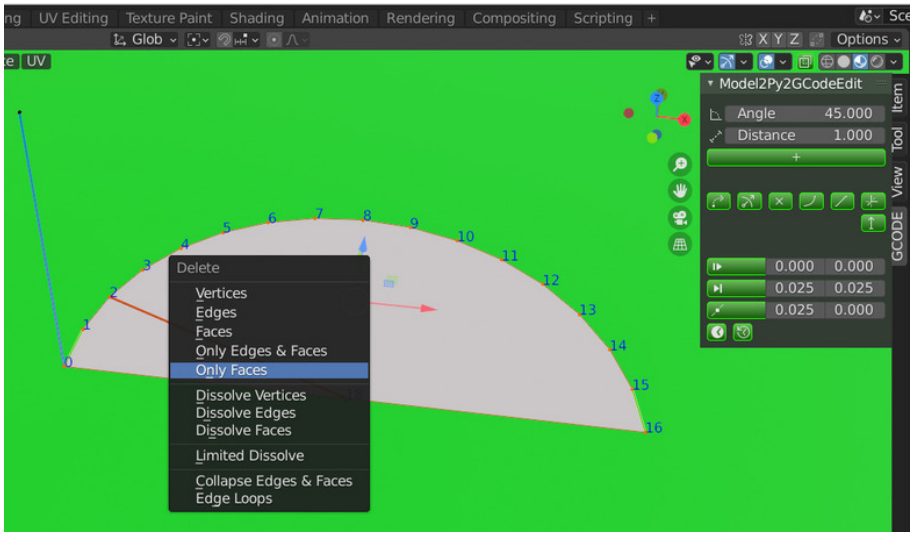

Figure 284: To remove a face, select all the vertices of that face then press "Delete" and select "Faces Only". (And yes, for those who are more knowledgeable on Blender there is a way to select a face with a single click but that is not shown here.)

1.10.3 Joining Paths and Meshes

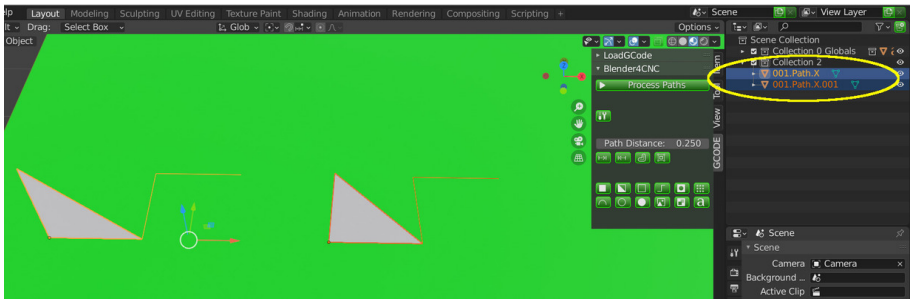

Figure 285: Sometimes it is useful to join meshes together to create your desired result. Here we have two paths. First, select the path on the right, then shift-click to also select the path on the right (selecting in this order so that the left path is the last to be selected will result in the origin of the joined path being at the lower left corner -not necessary but convenient).

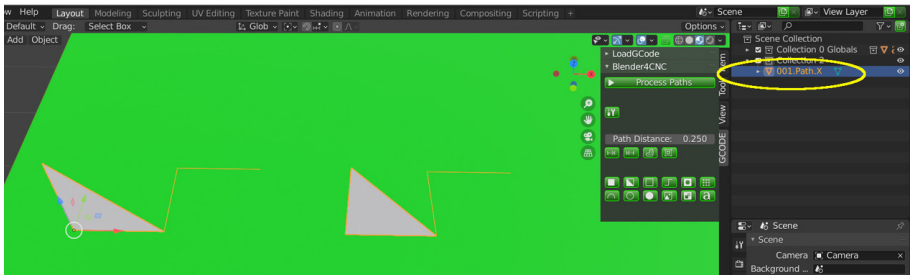

Figure 286: Press ctrl-j to join the paths (notice the origin appears at the lower left as we expected). You can see in the list of objects that we now have only one path. However, the path is disjointed and has 2 start points.

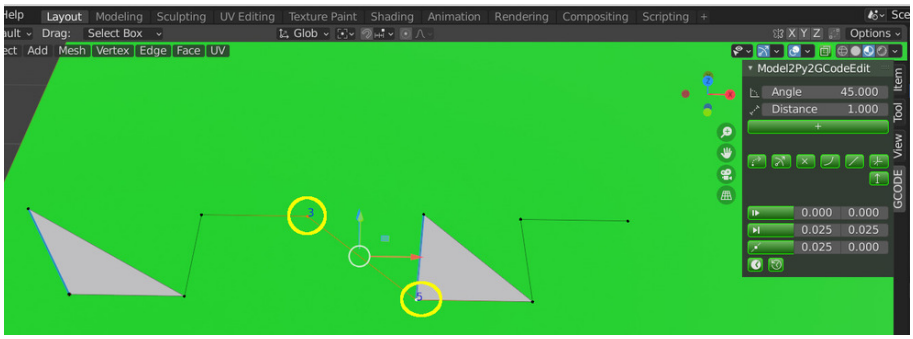

Figure 287: Press TAB to enter edit mode and select the two circled vertices. To create an edge between them press "f". Now the path is continuous.

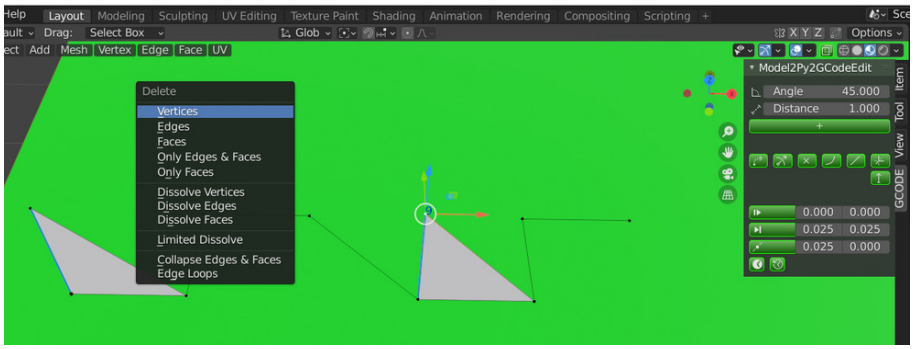

Figure 288: Select the top of the second start point and press "Delete", then select "Vertices" to remove this point.

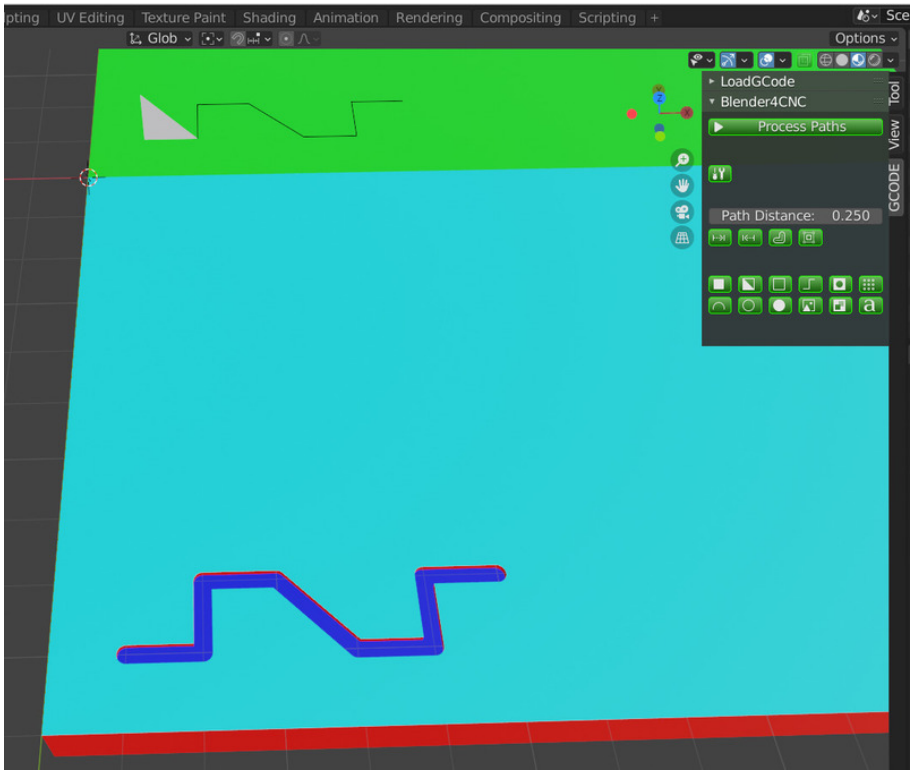

Figure 289: Press TAB to exit edit mode then click the "Process Paths" button. The two paths (two blender meshes) have been successfully joined together to create a single path operation.

1.10.4 Merging Vertices as a method of deleting vertices and corners

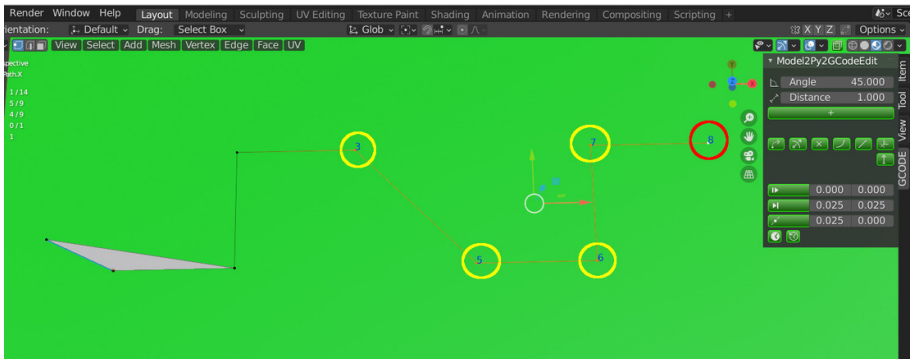

Figure 290: Deleting a vertex also deteles any edges attached to it. Sometimes, you want to remove a corner/vertex but have the edges snap to between the remaining vertices. Here we have a path in edit mode and we are going to eliminate a couple of corners at one time. Select the marked vertices but make sure you shift-click the point marked in red LAST.

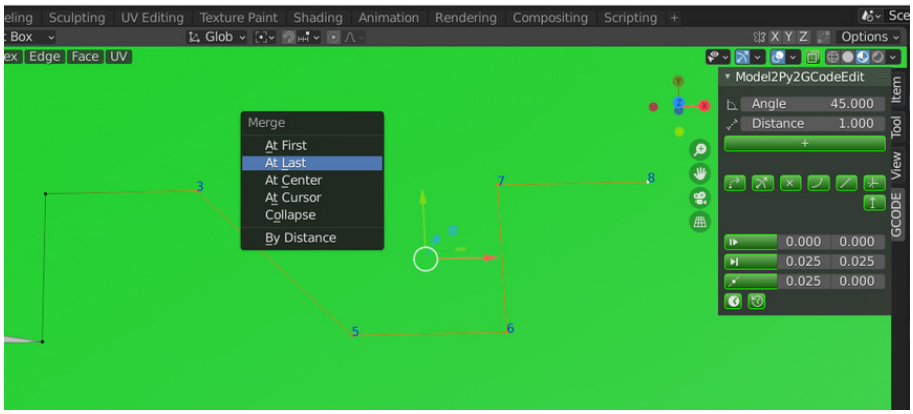

Figure 291: Press "m" and select "At Last".

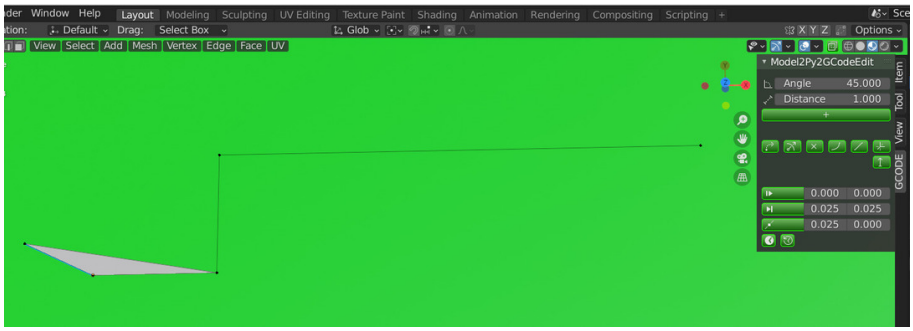

Figure 292: The selected vertices have all been merged with the last selected vertex leaving us with a long straight line in the path.

1.10.5 Splitting an edge to create a vertex or corner

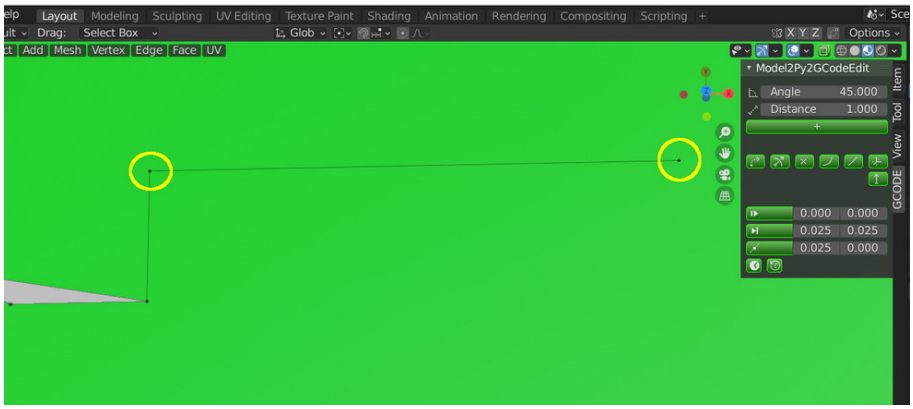

Figure 293: Let's say we want to add a vertex/corner between the selected vertices.

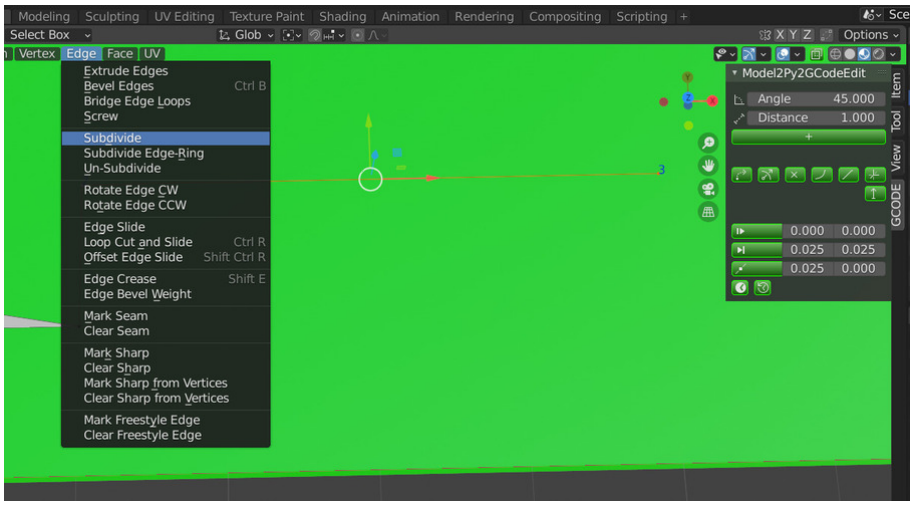

Figure 294: Select "Subdivide" from the Edge menu.

Figure 295: A new vertex appears halfway along the edge which you can then drag to create a corner.

1.10.6 Adding a point to an operation

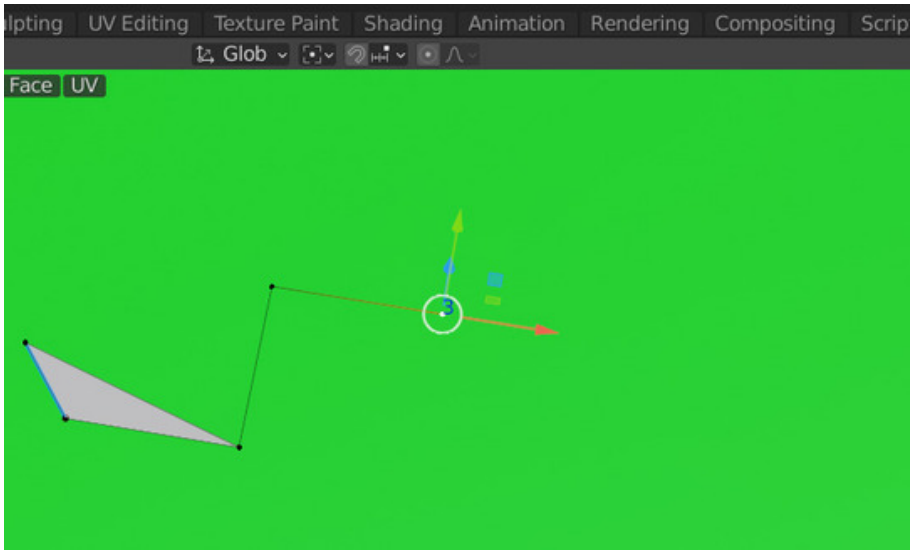

Figure 296: Let's add a point to this path. Select the last vertex (you could select any vertex if you want).

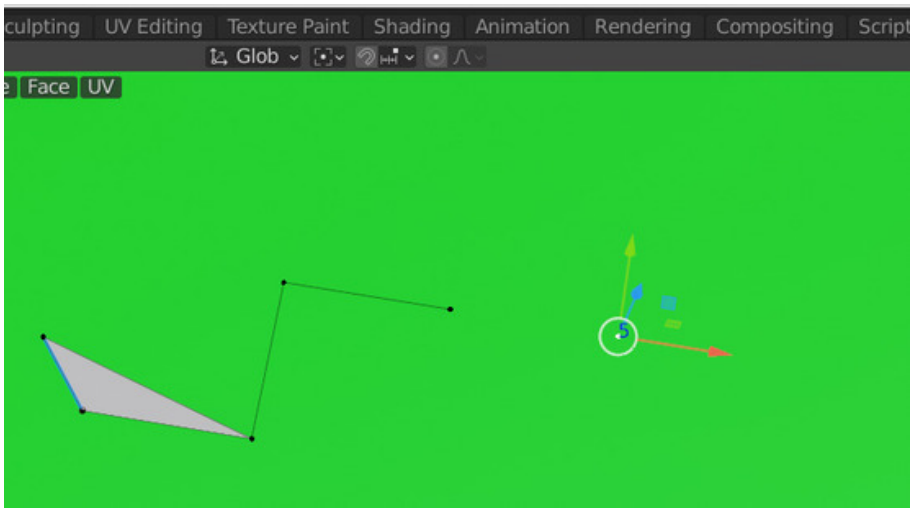

Figure 297: Duplicate the vertex by pressing shift-d and then press enter. Now you can drag the new point to any location.

1.10.7 Simplify a Mesh - Limited Dissolve

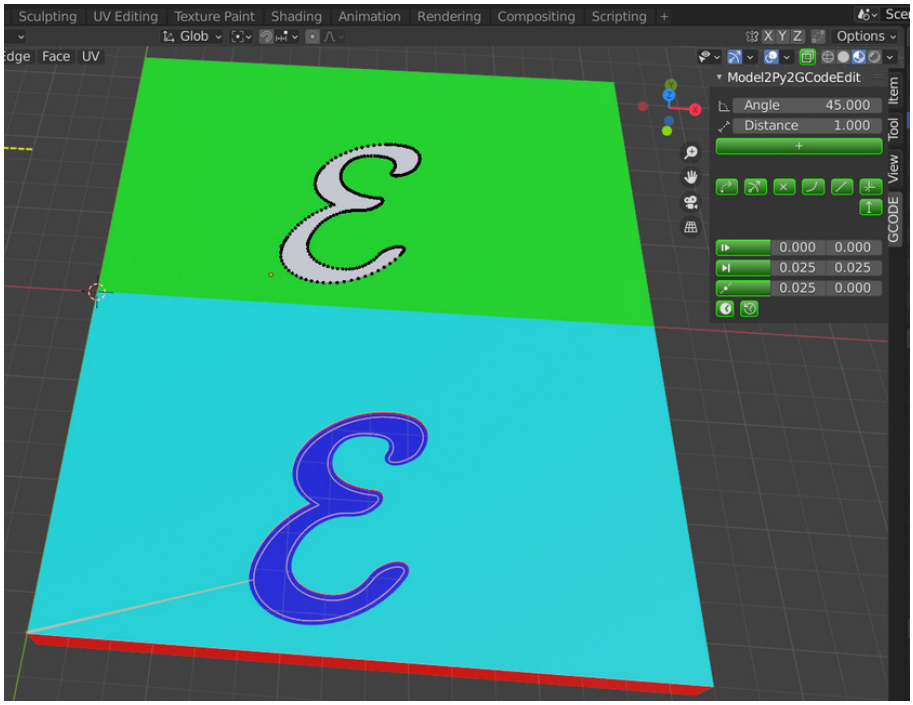

Figure 298: It takes time to process a mesh and produce GCode. The more vertices in a path, the longer it takes to process. More vertices can sometimes slow down GCode operation if it causes lots of accelerations and decelerations in the CNC machine. Therefore it is efficient to only use as many vertices as is necessary. Here is a project where the letter "E" (in a "MathJax" font in Linux) was converted into a mesh and then a CNC operation. This mesh has 324 vertices and at the time of this writing, it took 1 minute and 51 seconds for Blender4CNC to visualize it and produce the GCode (visualizing takes the most time).

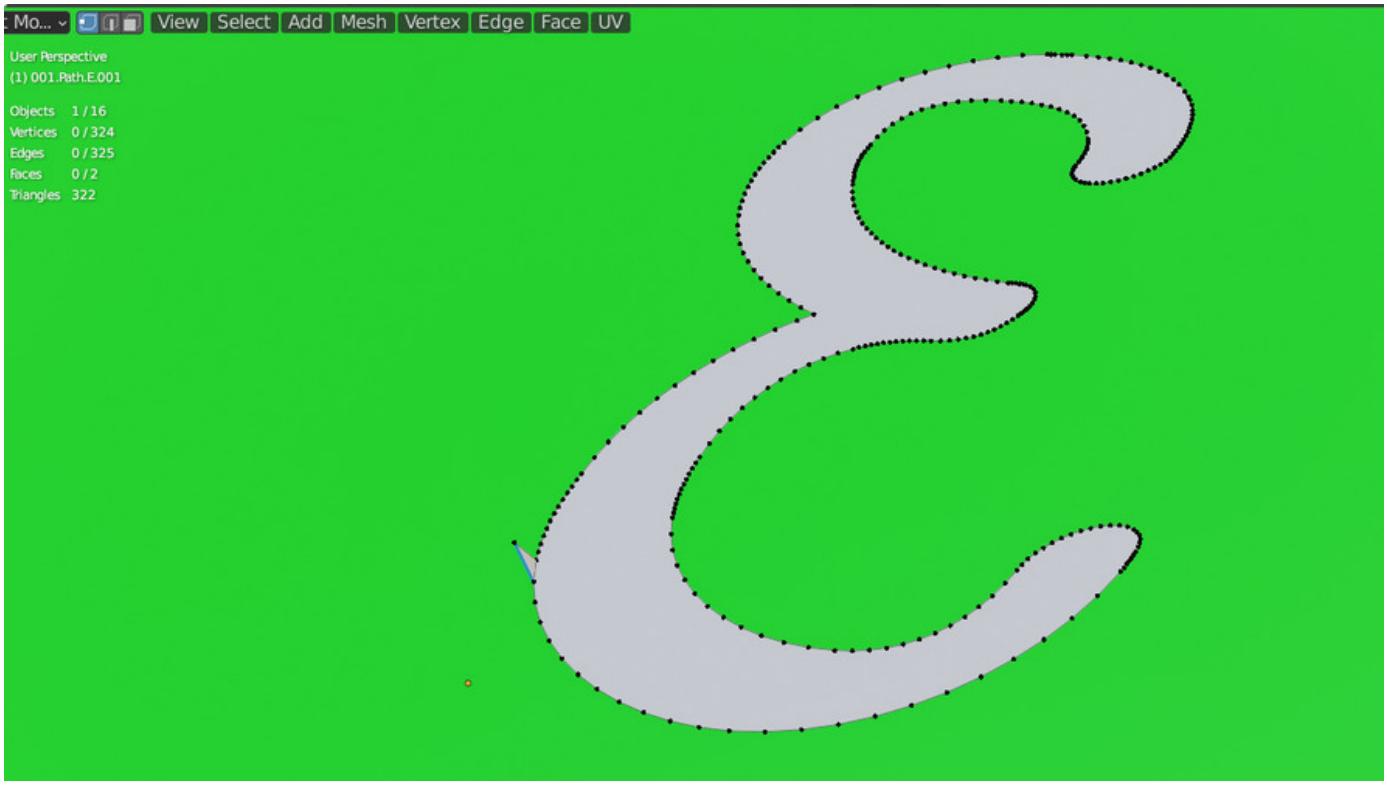

Figure 299: If we select the path, press TAB for edit mode and then zoom in, we can see that this path is made up of 324 vertices. Is this level of detail really going to be visible in the finished product? Not in this case. Let's reduce the number of vertices without changing the shape significantly.

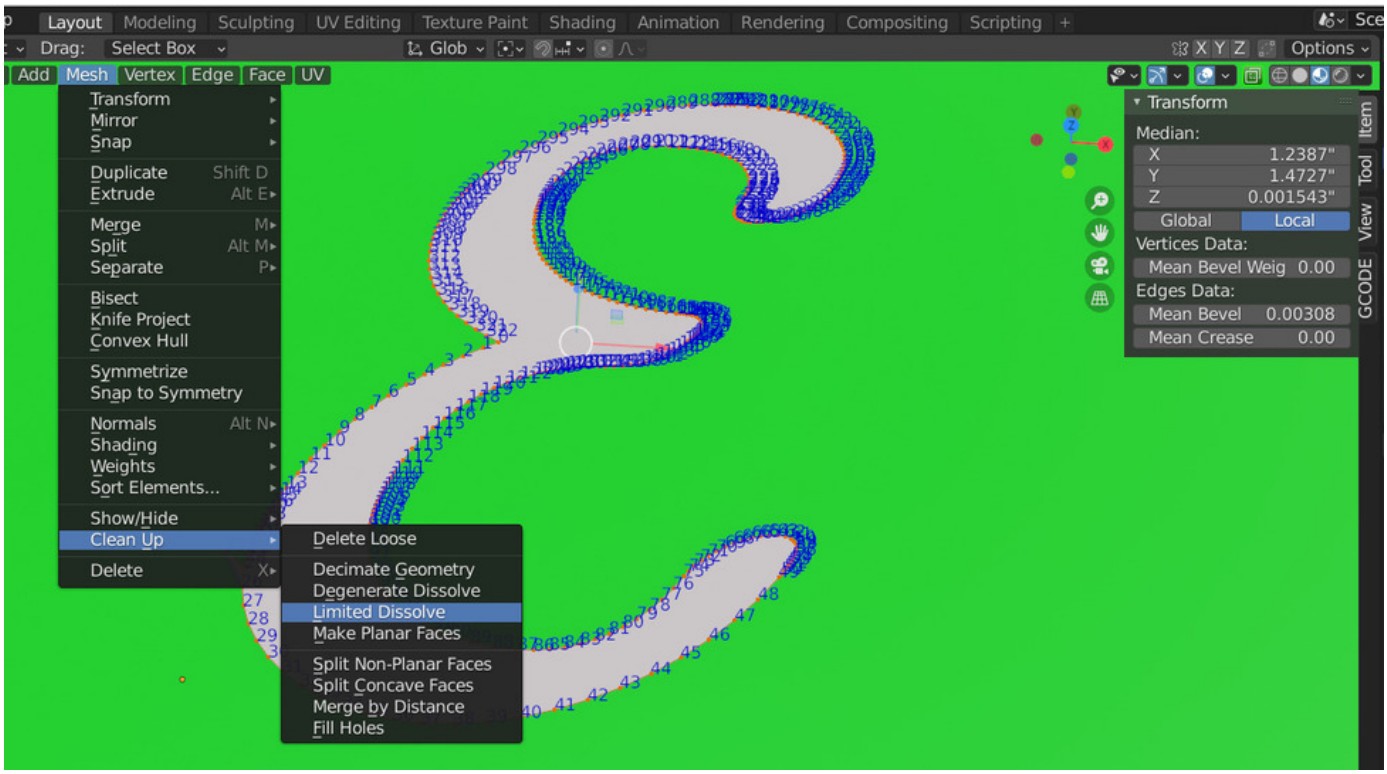

Figure 300: Select all the points (also when in edit mode you can select all points by pressing "a"). Then select "Mesh", "Clean Up" then "Limited Dissolve".

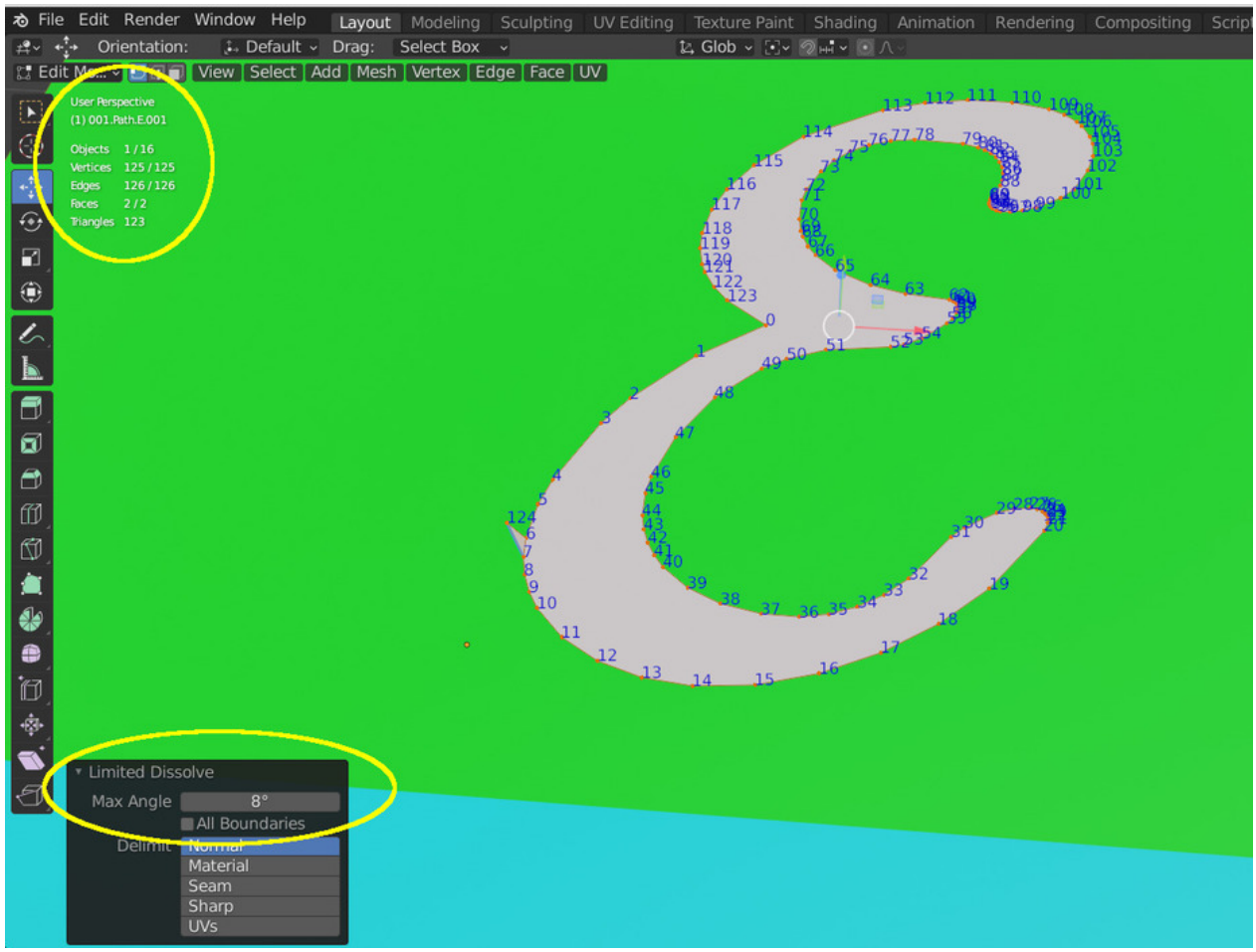

Figure 301: A small popup menu appears in the lower left of the viewport. The "Limited Dissolve" function will remove neighbor points that are within a certain angle and this is useful for reducing points around curved paths. By increasing the "dissolve" angle to 8 degrees, the number of vertices is reduced to 125. (Just make sure you don't accidently remove a vertice around the "start" flag.)

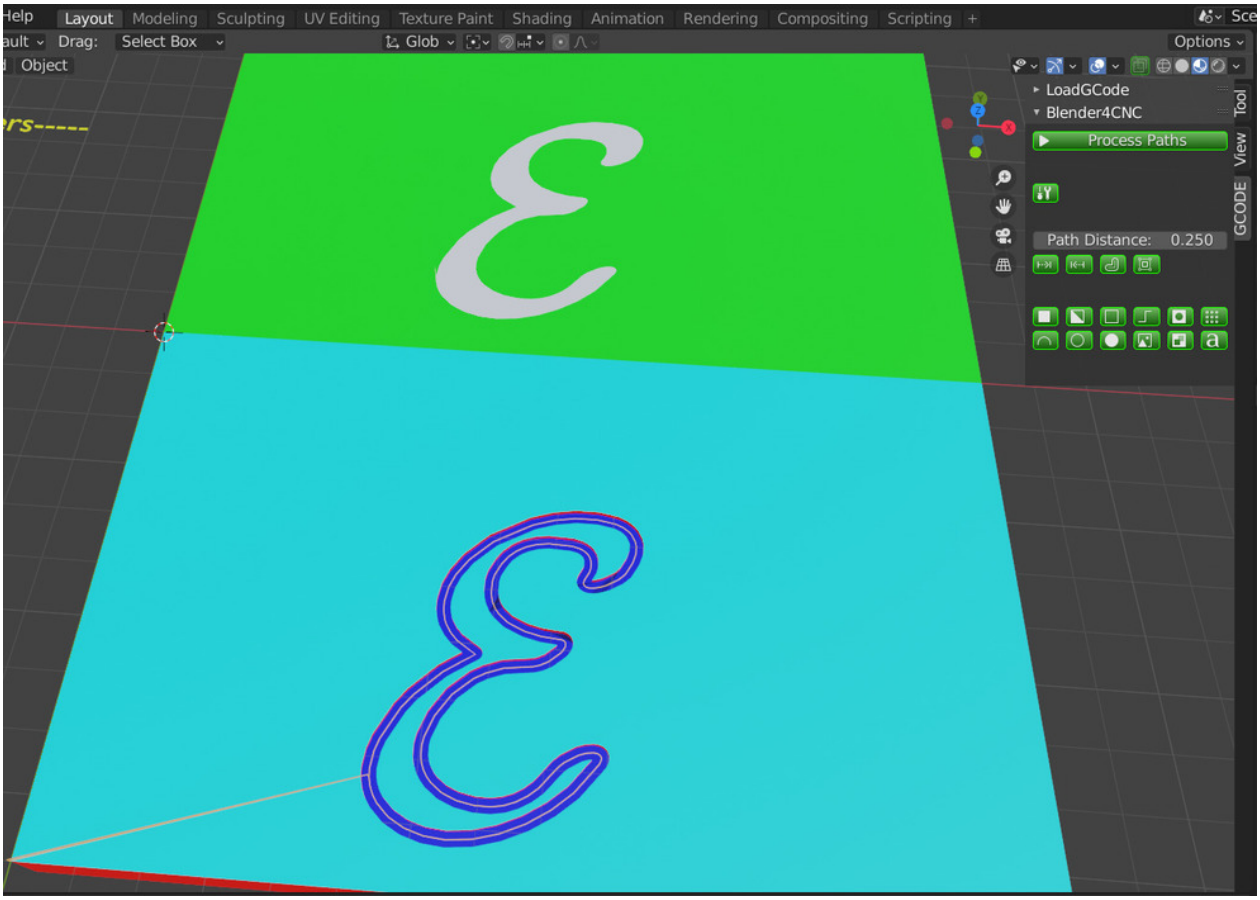

Figure 302: By pressing TAB to exit edit mode and then clicking "Process Paths" - this now only takes 14 seconds to visualize. If you look very closely you can see some straight segments around the curves but again, this is likely unnoticeable in the finished product.

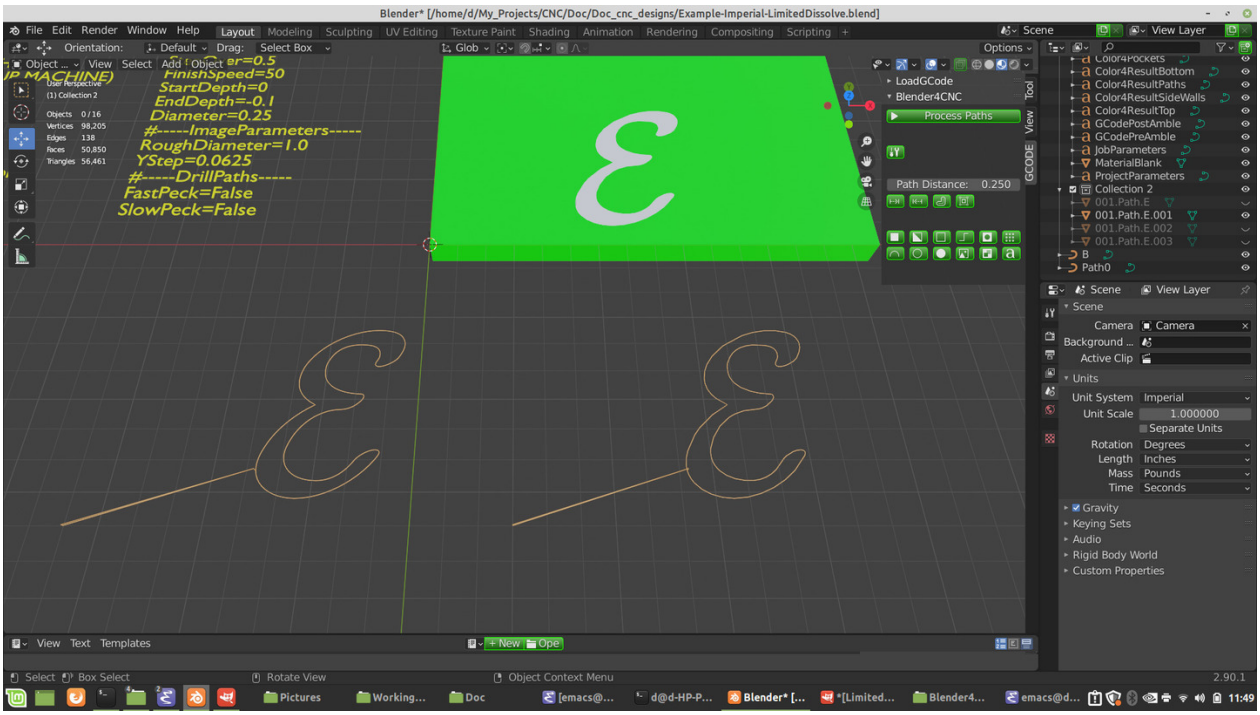

Figure 303: Here is a comparison between the original (left - 1 minute, 51 seconds) and a limited dissolve at 8 degrees and 125 points (right - 14 seconds).

1.10.8 Simplify a Mesh - Merge by Distance

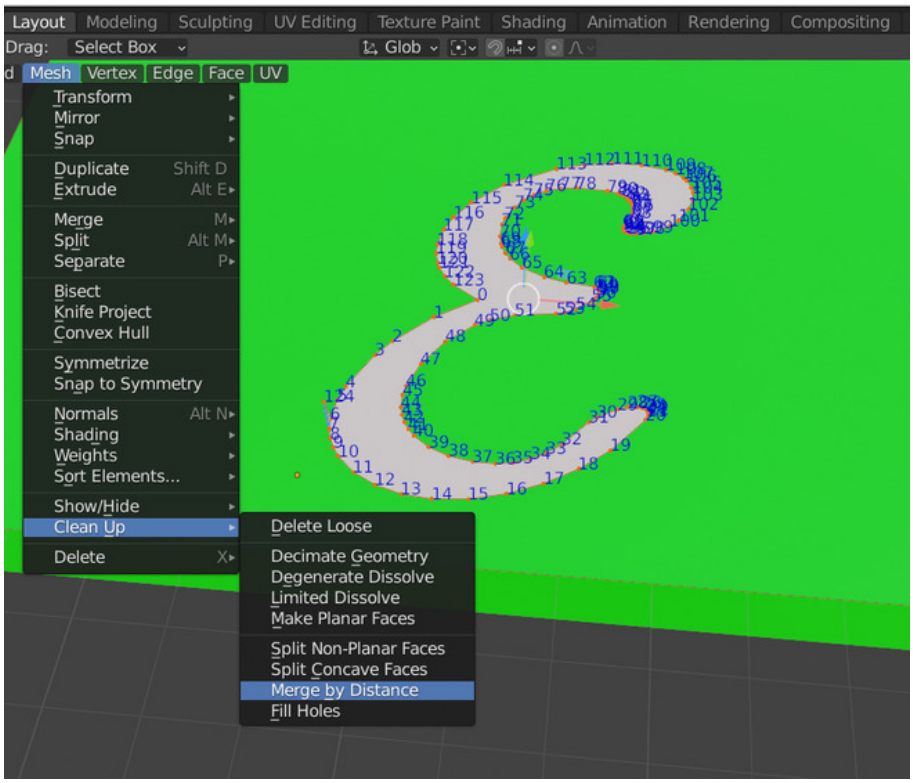

Figure 304: Similar to "Limited Dissolve" is the "Merge by Distance" function. When in edit mode, go to "Mesh", "Clean Up" and then "Merge by Distance".

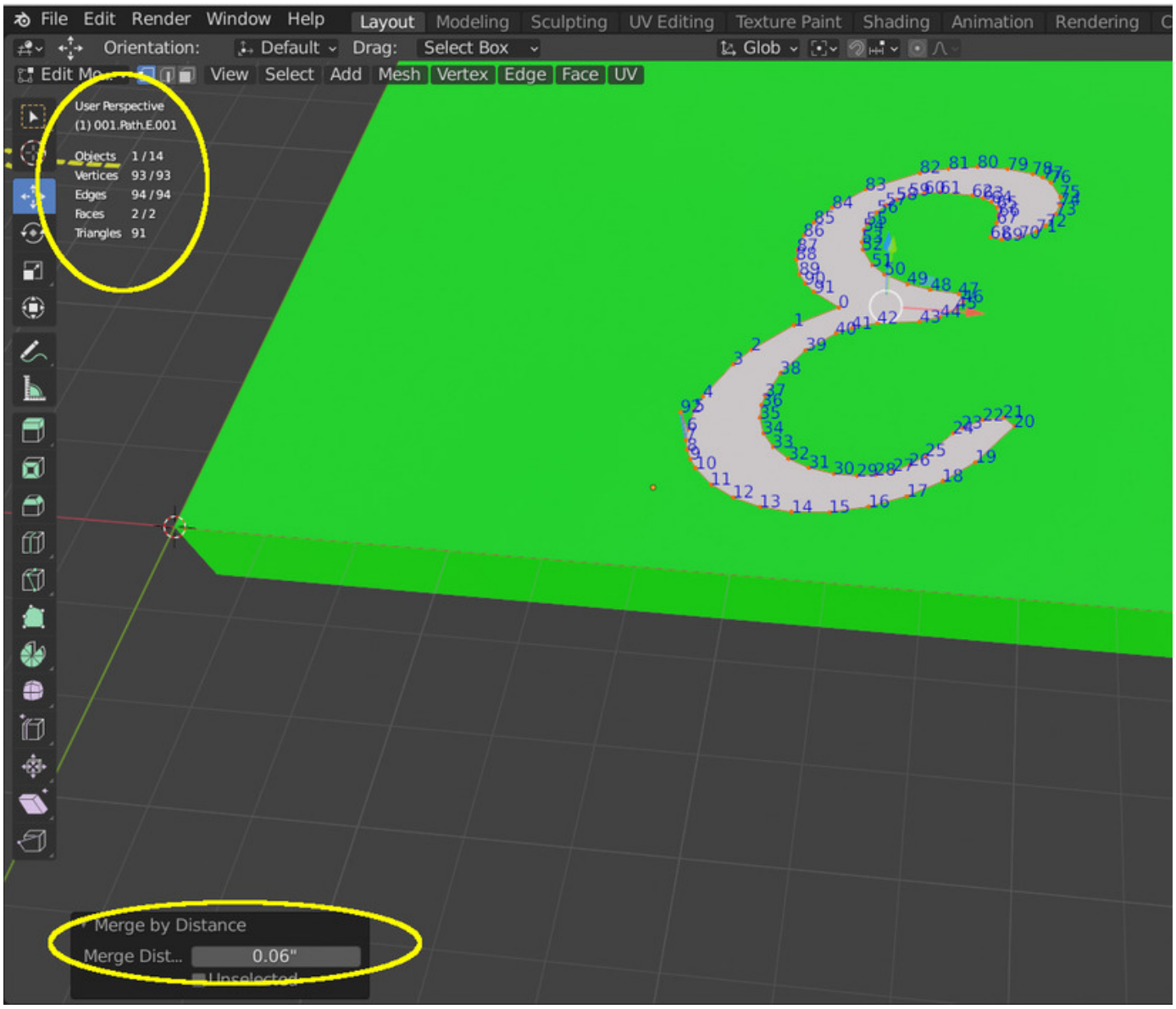

Figure 305: Again, a popup menu panel appears in the lower left of the viewport and by setting the distance to 0.060, the mesh is reduced to 93 points.

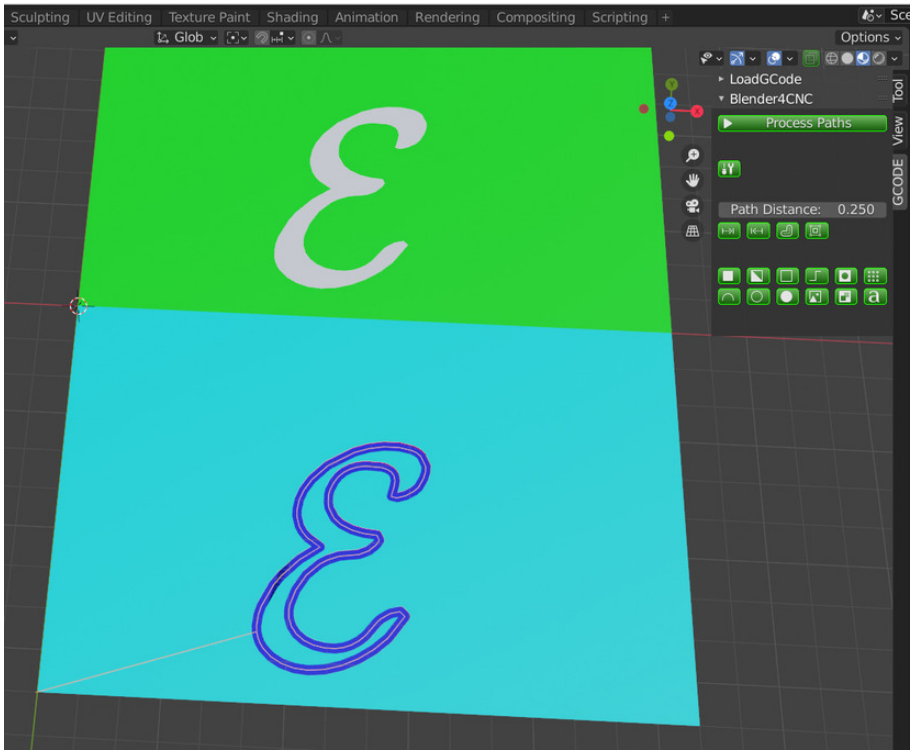

Figure 306: This takes about 7.8 seconds to visualize. Compared to "Limited Dissolve" this has smoother results on the larger curves but is less smooth on the sharper curves (again, these details may not be seen in the finished product).

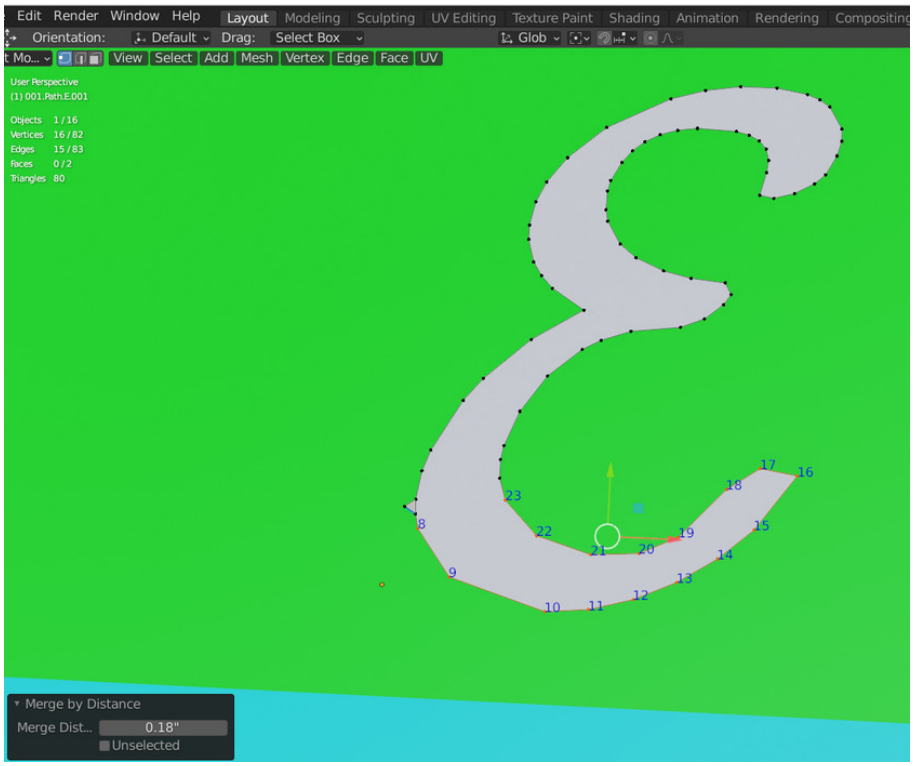

Figure 307: It is possible to apply "Merge by Distance" or "Limited Dissolve" to just some of the points. Just select the relevant points before selecting the function. In this case the lower part of the "E" was selected. The upper half of the "E" still has all the original points; the lower half has been drastically reduced. The reason why this letter "E" mesh had so many vertices in the first place is because when Blender converts an object to a mesh it approximates curves with linear segments. If you really want to reduce the number of vertices in a curvy operation, designing with arcs is very helpful - requires far fewer points and provides smooth results.

1.10.9 Touching Vertices in a Path

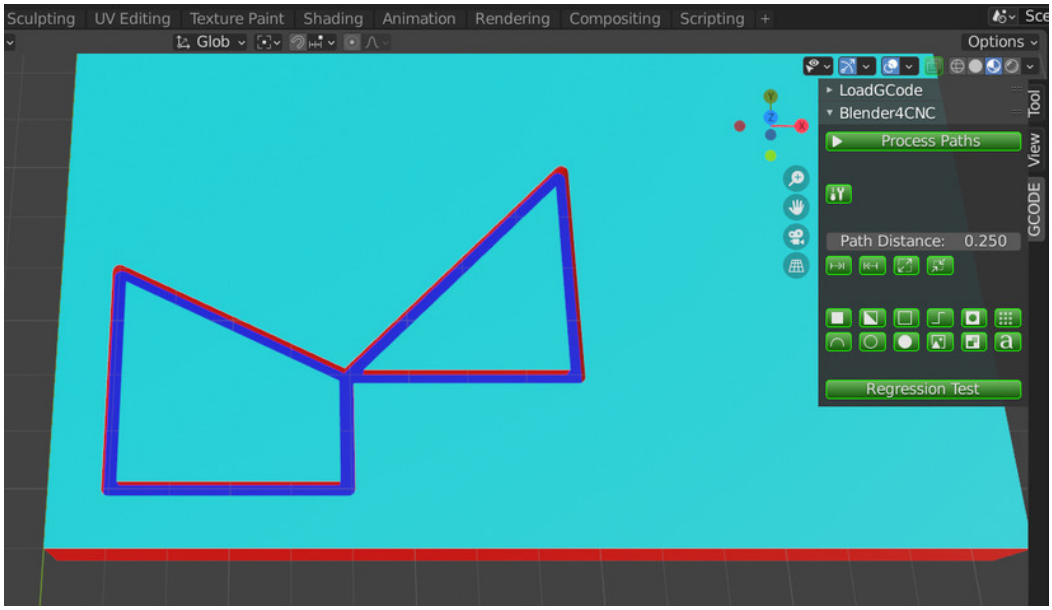

Figure 308: Sometimes you may wish to make a single, complex shape where a vertex gets "used" for connecting more than just two edges. Here is a shape that can demonstrate this case.

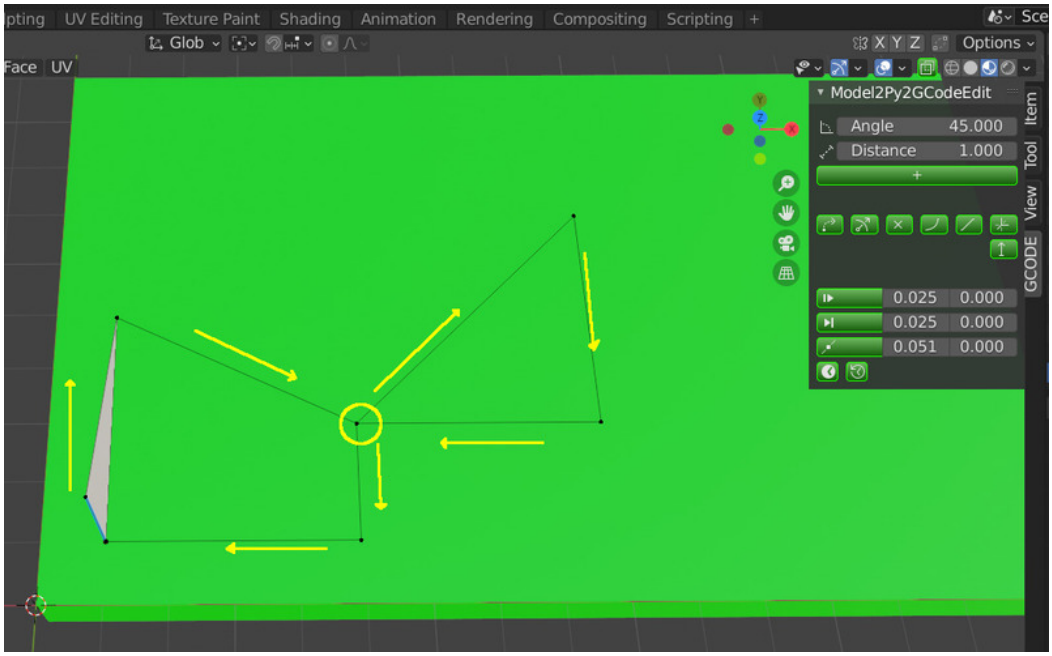

Figure 309: Here is a path that has been created to produce the desired shape. The intention is that the router bit should follow the directions shown by the yellow lines and you can see how the movement has uses the vertex in the middle "twice". The cutter is to start in the lower left corner and move "up" and then move down and to the right to reach the center point. Then it moves up and to the right; then down; then to the left to reach the center point again. (Then it moves down and then turns to the left to finish where it started. But watch what happens if we click "Process Paths".

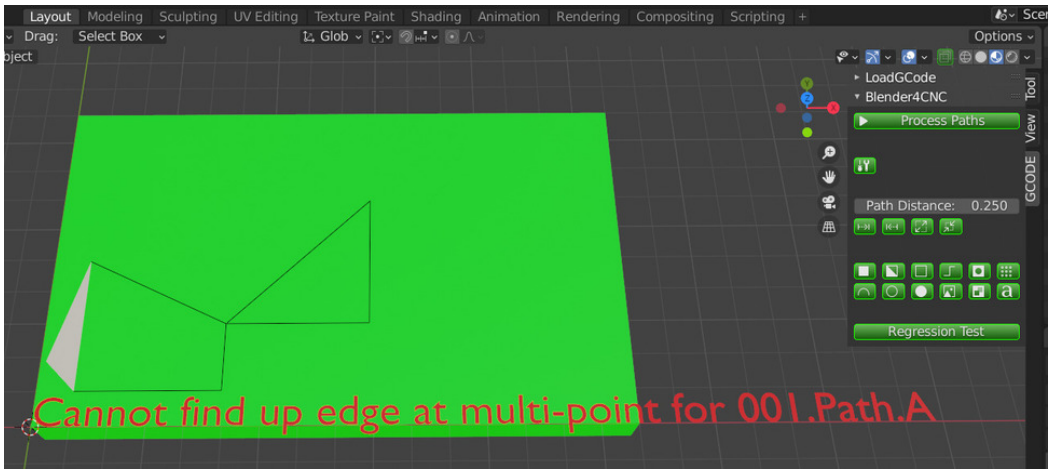

Figure 310: We get an error! This is because a vertex can only be used to connect two edges (not four edges like in this example. There are two ways to achieve this shape (without resorting to breaking it into two separate operations).

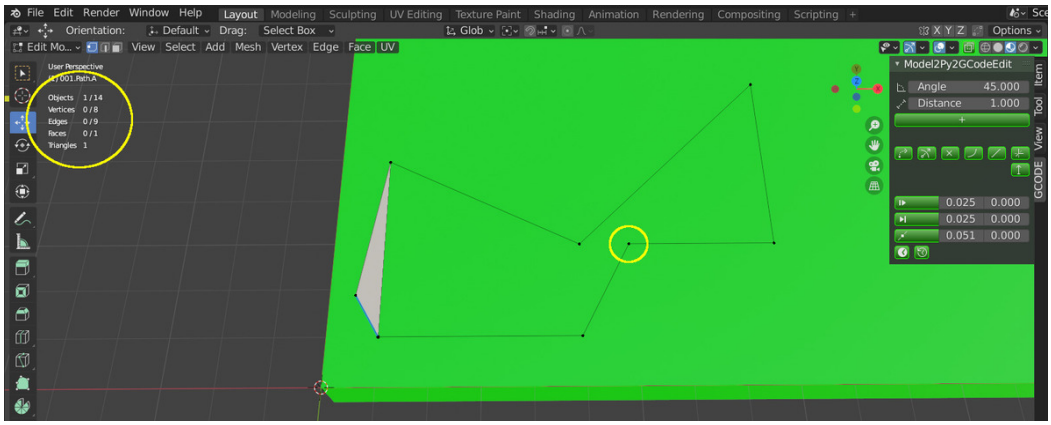

Figure 311: The first approach is to actually use **two** vertices at the same location. Create the path as a single loop as shown and observe that there are **8** vertices in the shape whereas there was previously only 7 vertices.

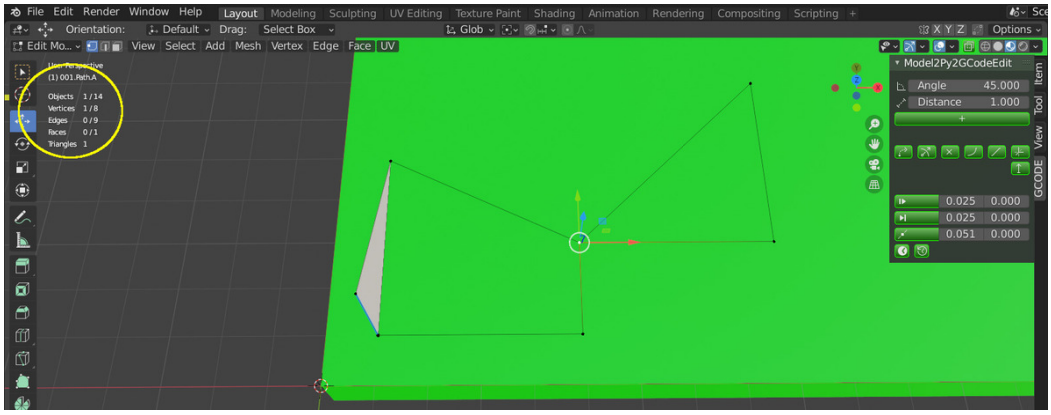

Figure 312: Then move the 2nd vertex to the exact location of the first vertex in the center. It looks like there is only one vertex in the center but there are actually two vertices and note that the statistics still show that there are 8 vertices in the mesh (even though we can only see 7).

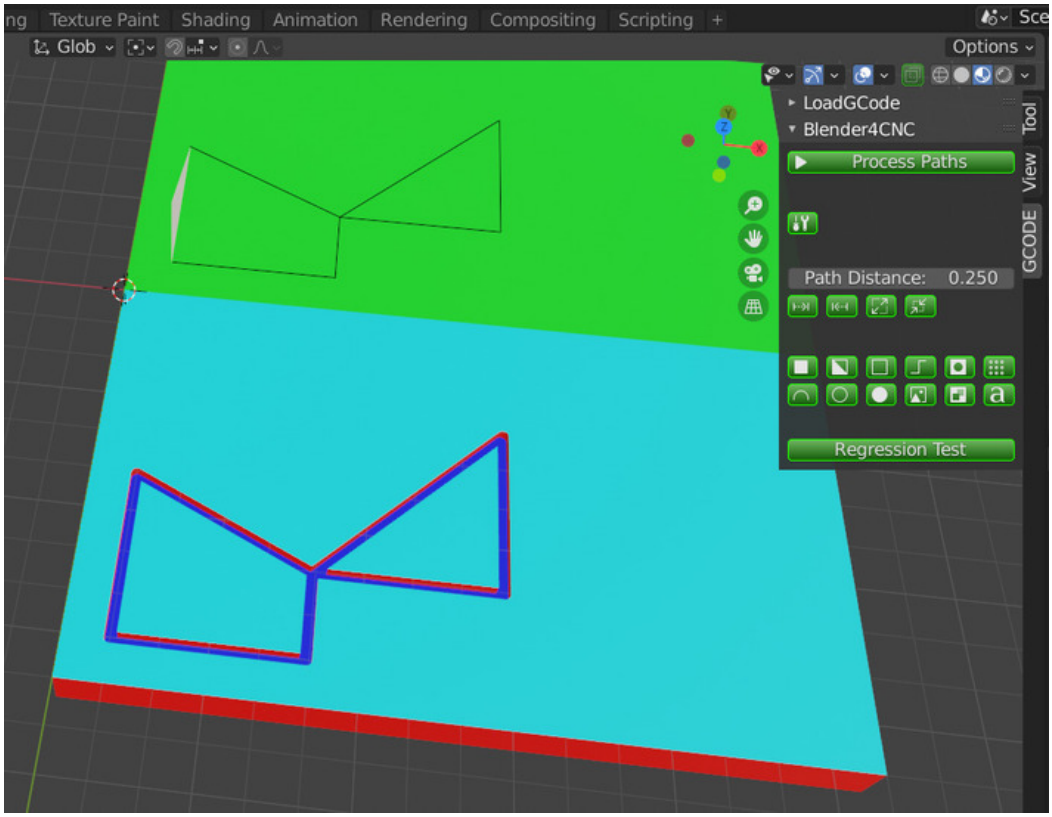

Figure 313: This works and produces the desired shape. However, it can be frustrating to deal with a mesh where multiple points are directly overlaying each other. Just figuring out how to select one of the points and not the other is difficult. So there is an alternative method of dealing with this case.

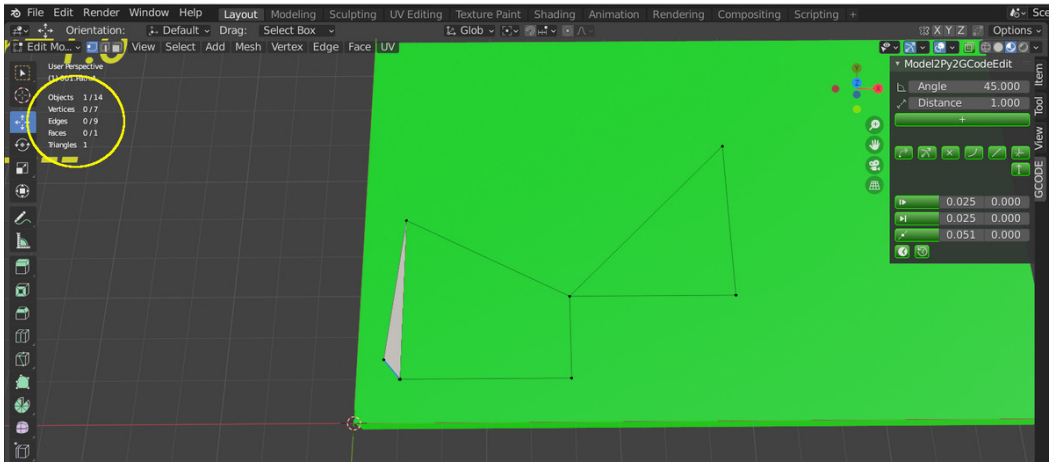

Figure 314: Go back to the first path that produced the error. Note the statistics show we only have 7 vertices so we know there is only one vertex in the center and it is connected to 4 edges.

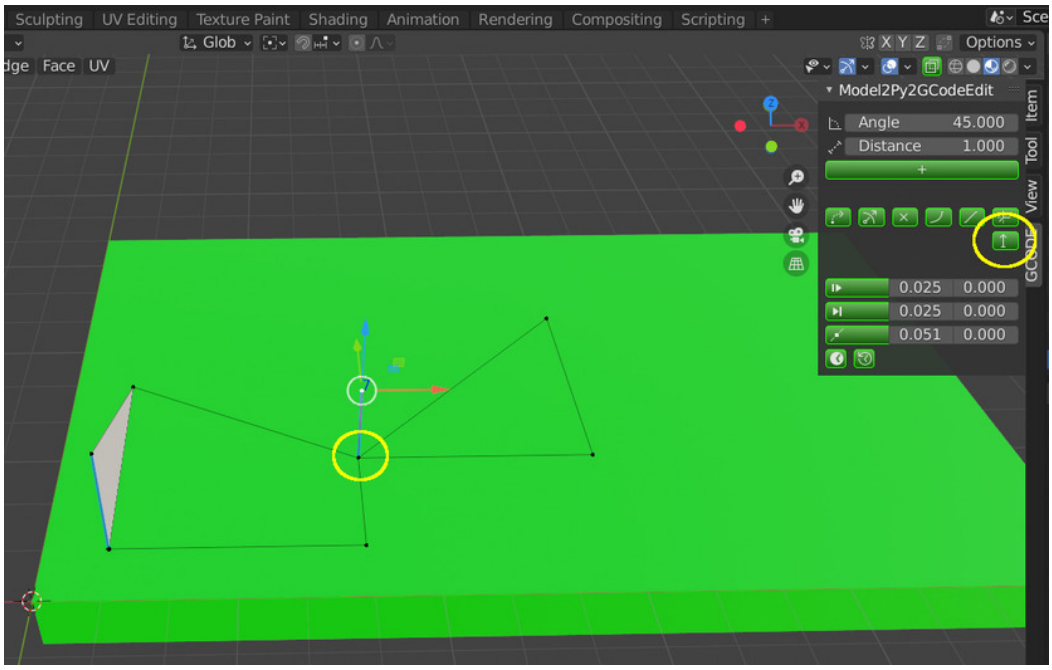

Figure 315: We need to create a "direction flag" by creating a vertex directly above the center vertex. The easiest way to do this is to select the center vertex, then click the "Make Up Point" button. You will see a new vertex above the original and the ege **looks** like a start point (although it is not - it is just an "up" point).

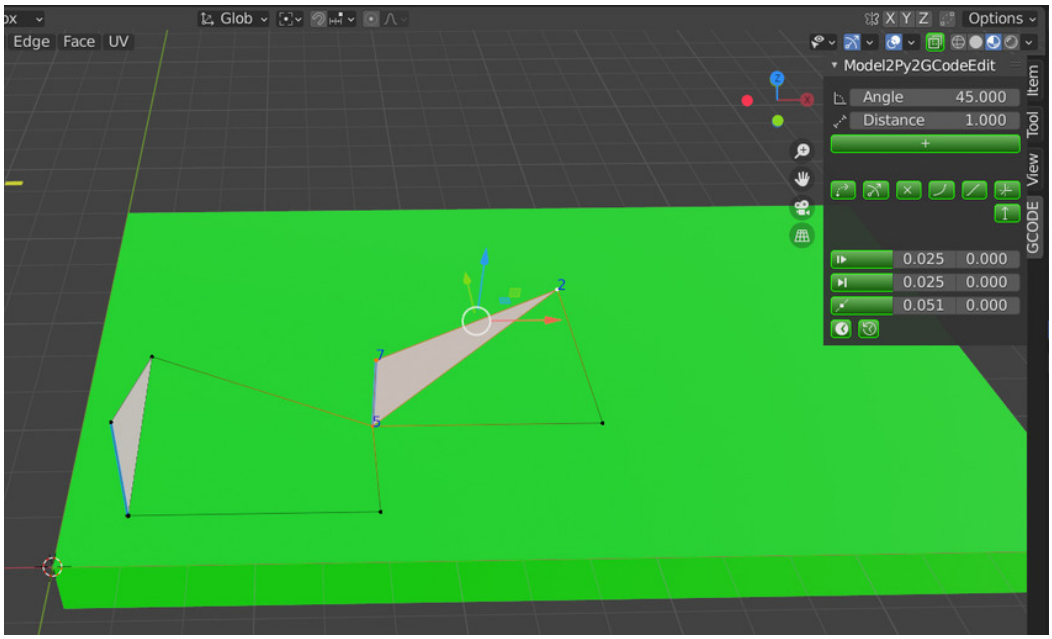

Figure 316: Now select the center vertex, the new vertex above, and the 3rd vertex in the upper right. Press 'f' to create the "direction flag". When Blender4CNC is processing a path it looks for points like this which are connected to more than two edges. When it finds such a point, it looks for one of these direction flags to tell it which edge it should follow next. If it finds a flag like this, it will continue without error.

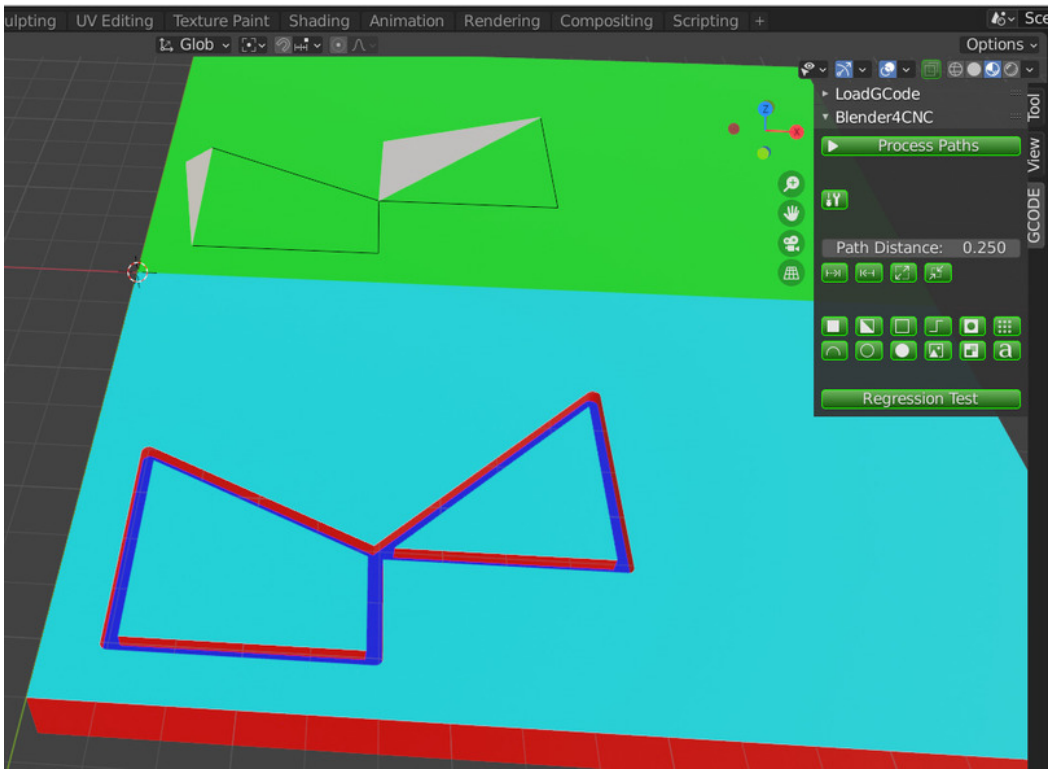

Figure 317: Blender4CNC processes the path without error.

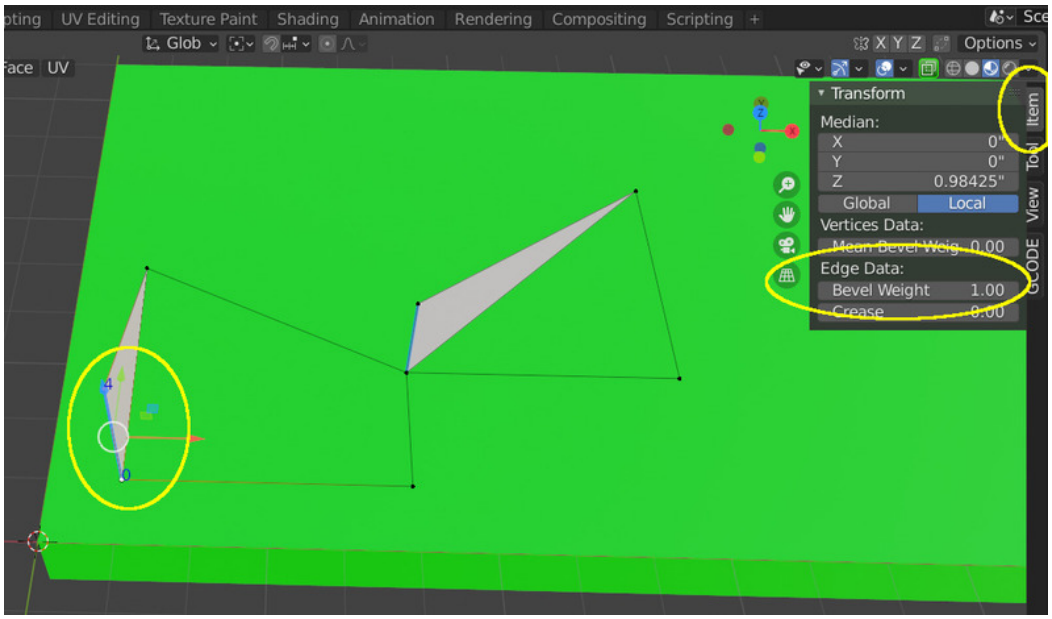

Figure 318: How can you tell the difference between a "Start" point and an "Up" point if you have created a complicated self-touching shape like this? The edge of a "Start" point has a Blender internal "Bevel Weight" of 1.0. Here we have selected the start point edge and you can see the Bevel Weight is set to 1.0.

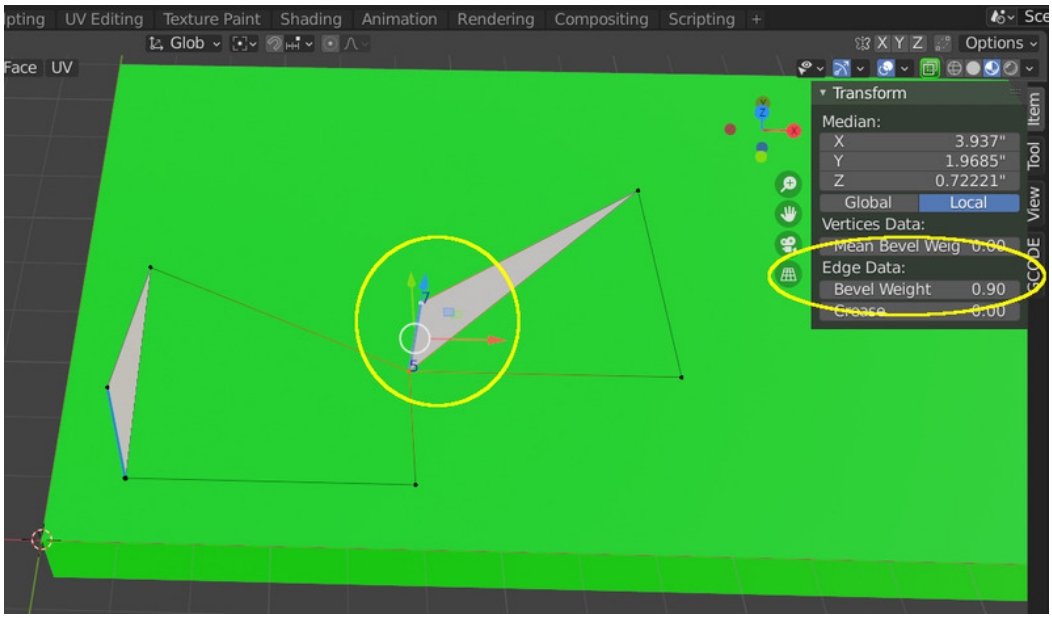

Figure 319: The edge of an "Up" point has a Blender internal "Bevel Weight" of 0.9. Here we have selected the "up" point edge and you can see the Bevel Weight is set to 0.9. That is the only difference between a "Start" point and an "Up" edge - one has a Bevel Weight of 1.0 and the other has a Bevel Weight of 0.9.

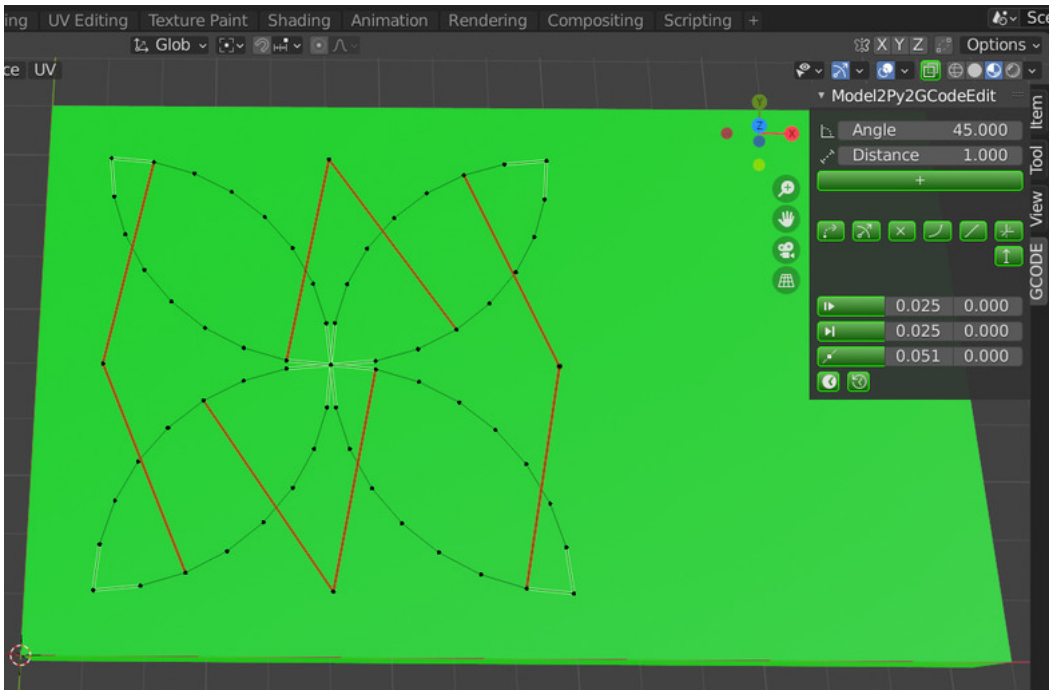

Figure 320: What if a shape gets even more complicatedand has a vertex that is used many times? Here is a path specifying a 4-petal flower made from 8 curves that start or end at the same center point. We want to start in the center and follow around each petal. This requires more than just a start point and a direction flag. We are going to have to create multiple "direction" flags and set a "Bevel Weight" on the "top" edge of the flags to indicate an ordering.

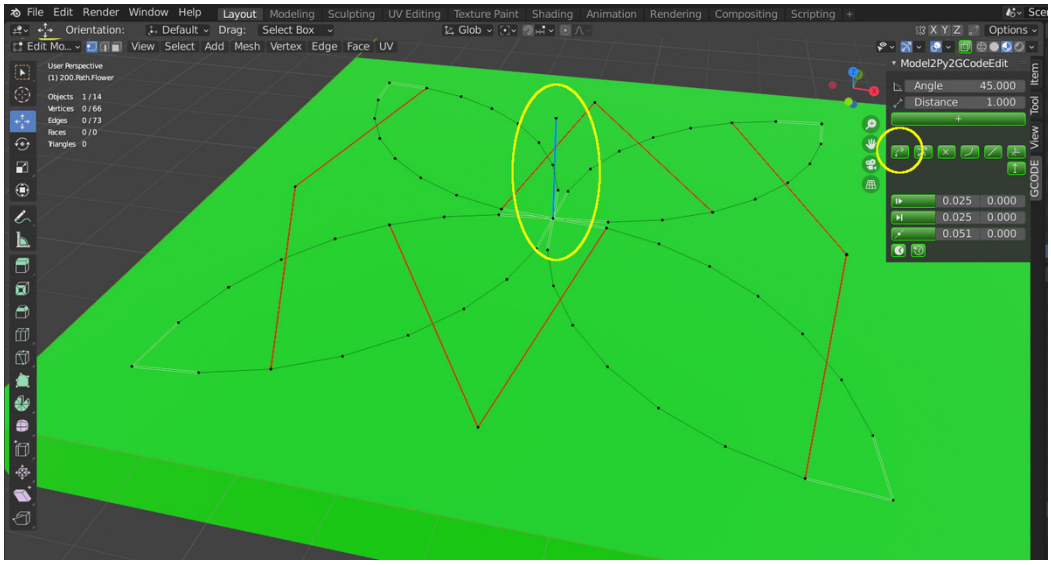

Figure 321: Select the center point and create a "Start" point.

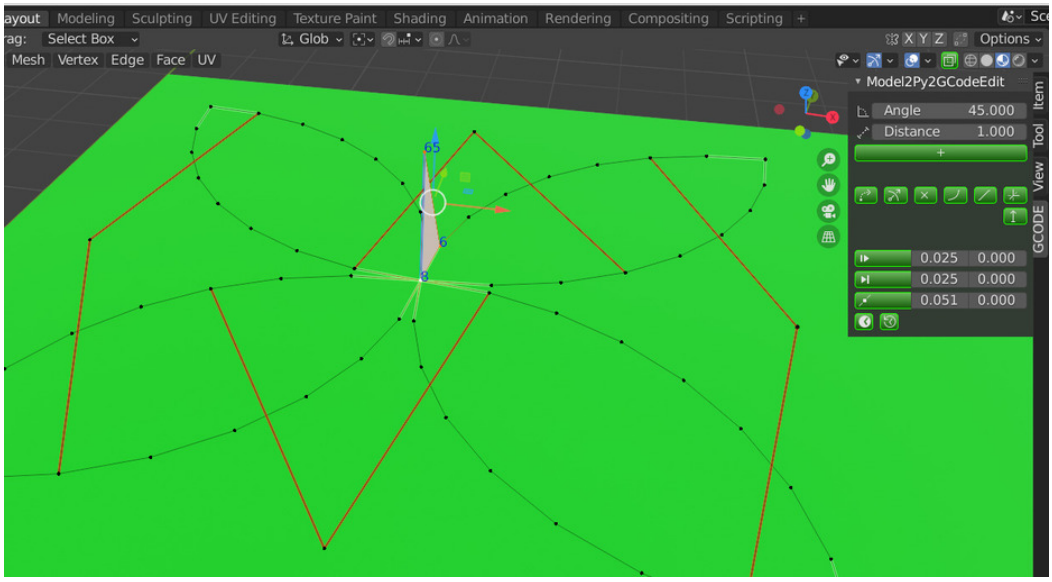

Figure 322: Add to the selection the fist point on the first petal and press "f" to create a triangular face.

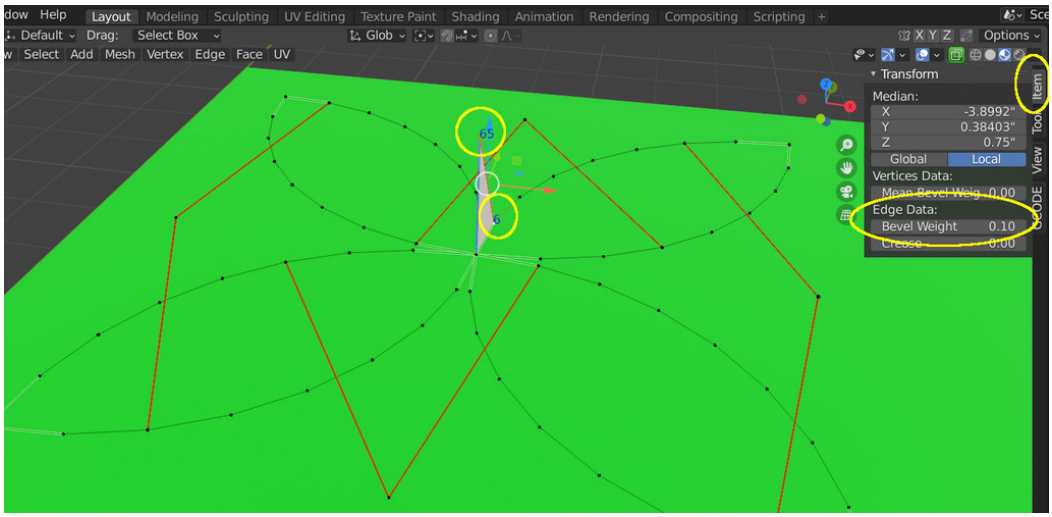

Figure 323: Select the "Start" point and the point on the petal so that the edge between them is selected. Set the Bevel Weight of this edge to 0.1.

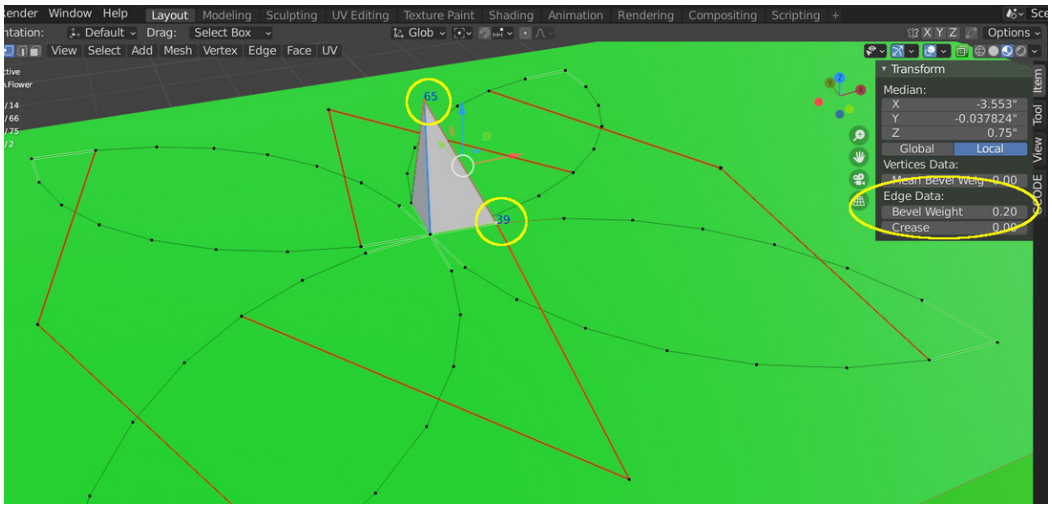

Figure 324: Repeat similarly for the 2nd petal. Select the center point, start point and the point on the petal and create a triangular face. Them select just the start point and the petal point so that the edge between them is selected. Set the Bevel Weight of this edge to 0.2.

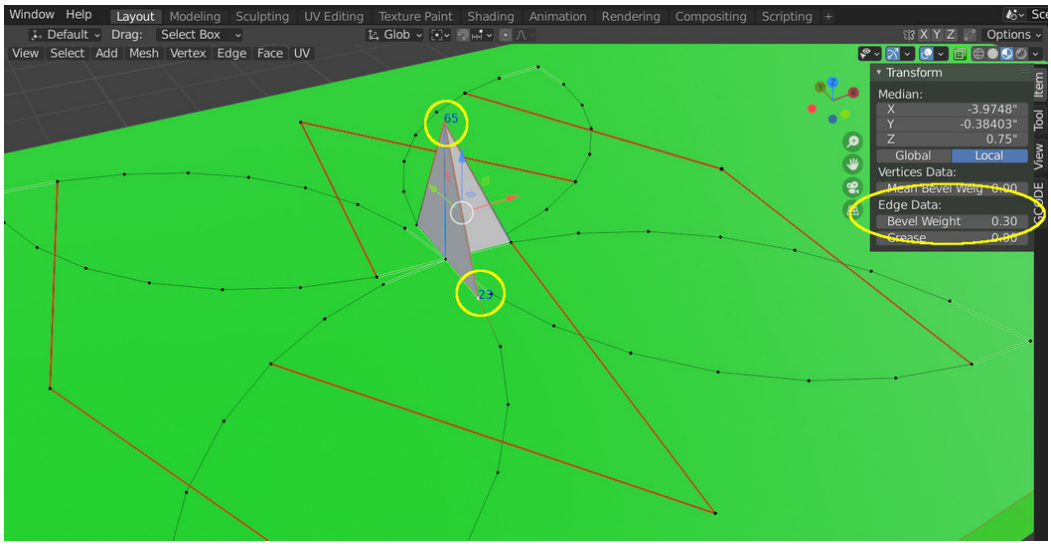

Figure 325: Repeat similarly for the 3rd petal. Set the Bevel Weight of this edge to 0.3.

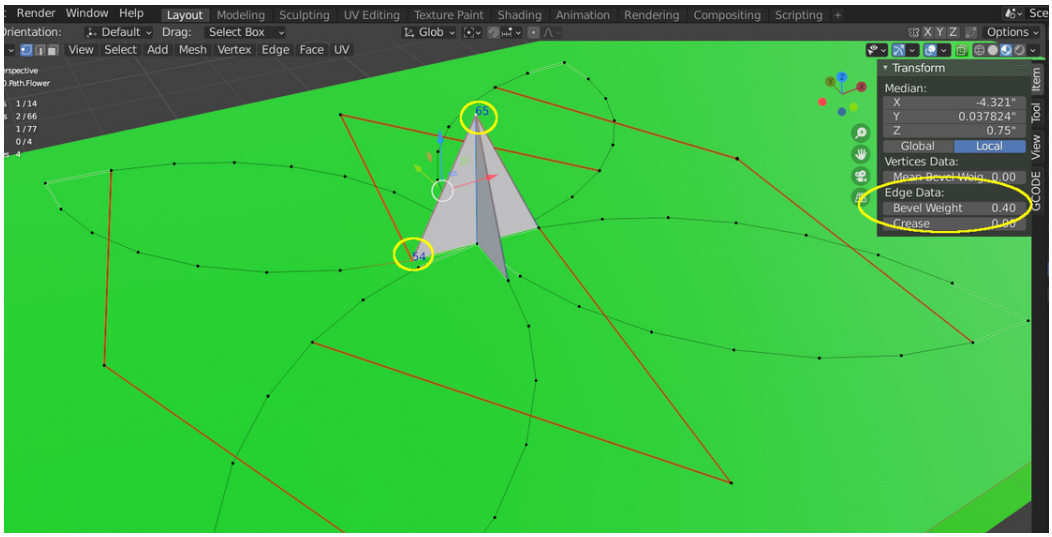

Figure 326: Repeat similarly for the 4th petal. Set the Bevel Weight of this edge to 0.4.

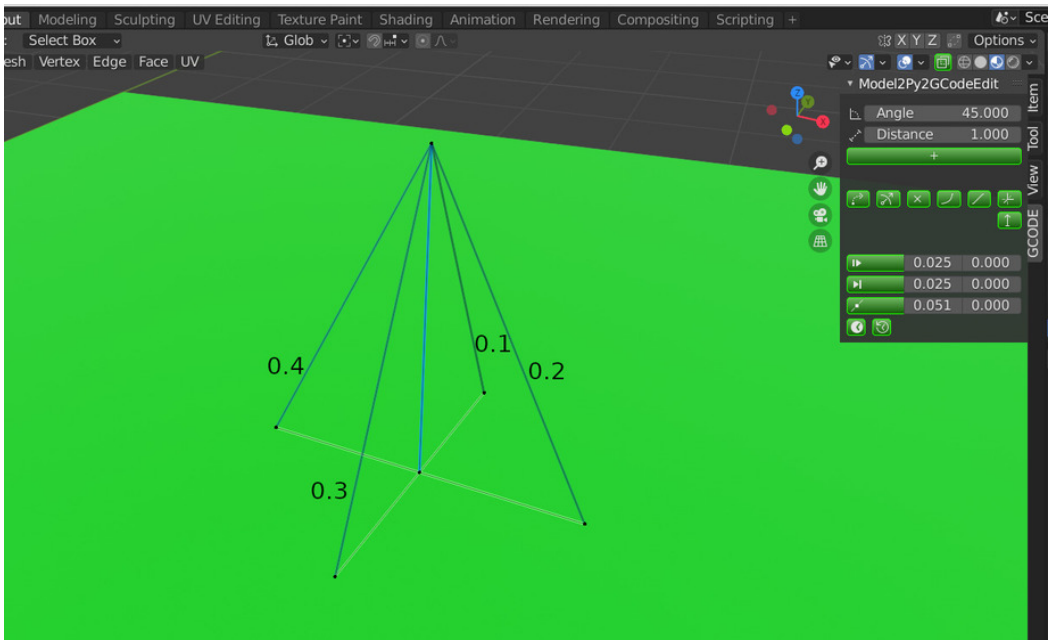

Figure 327: If we just examine the structure of the edges around the start point we can see that there are 4 sloping edges, each with a different Bevel Weight. When Blender4CNC is following this path, each time it reaches the cetner point, it will follow the next edge with the lowest weight. Therefore, it wil start the path at the cetner point and initially will follow the direction of the sloping edge with a Bevel Weight of 0.1. After cutting the first petal and returning to the center, it will then follow the sloping edge with a weight of 0.2 and cut the 2nd petal. It will then follow the sloping edge with 0.3 and finally will follow the sloping edge of 0.4 and cut the final petal.

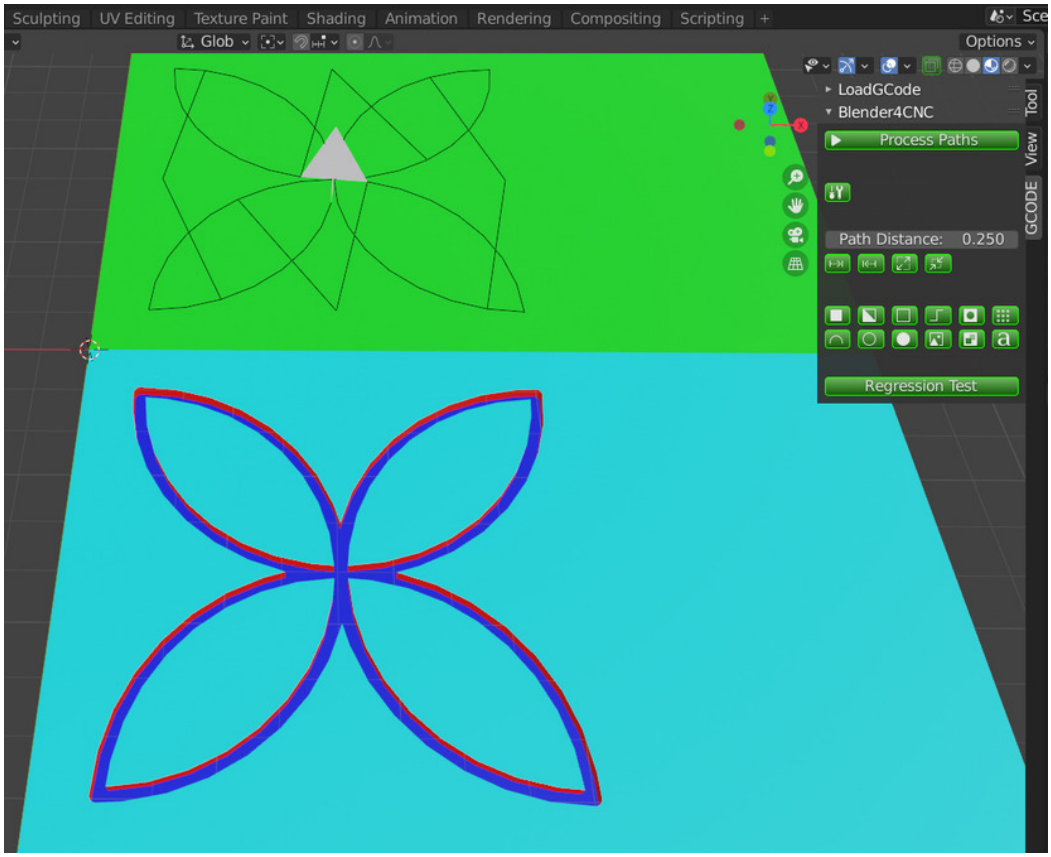

Figure 328: The path is cut properly.

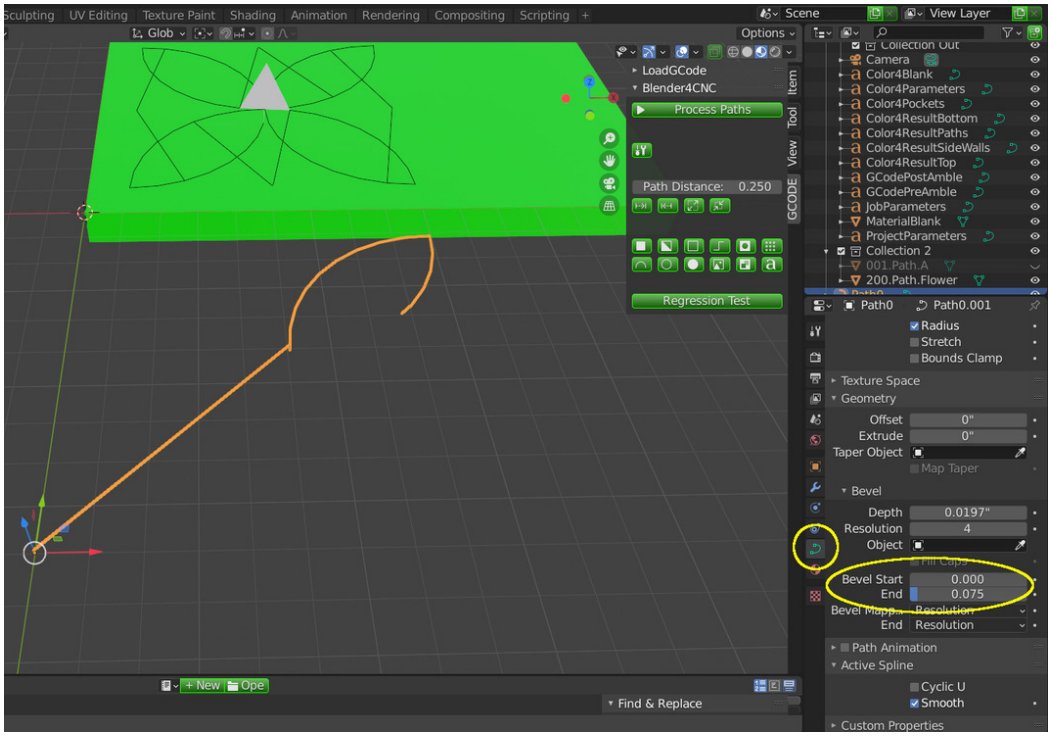

Figure 329: Here the block of material has been removed for clarity and we are just looking at the curve that represents the path of the cutter. By selecting the Curve tab and adjusting the "Bevel End" value 0.075 we can see how the cutter starts following the path from the center and around the first petal.

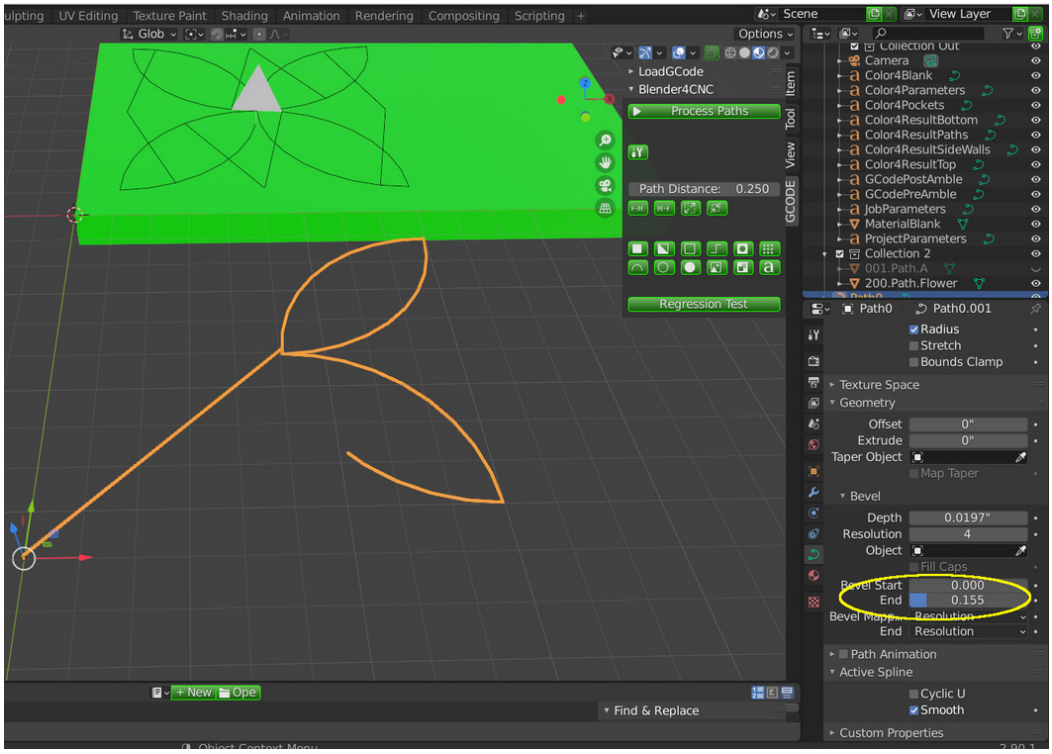

Figure 330: Then the cutter cuts the 2nd petal.

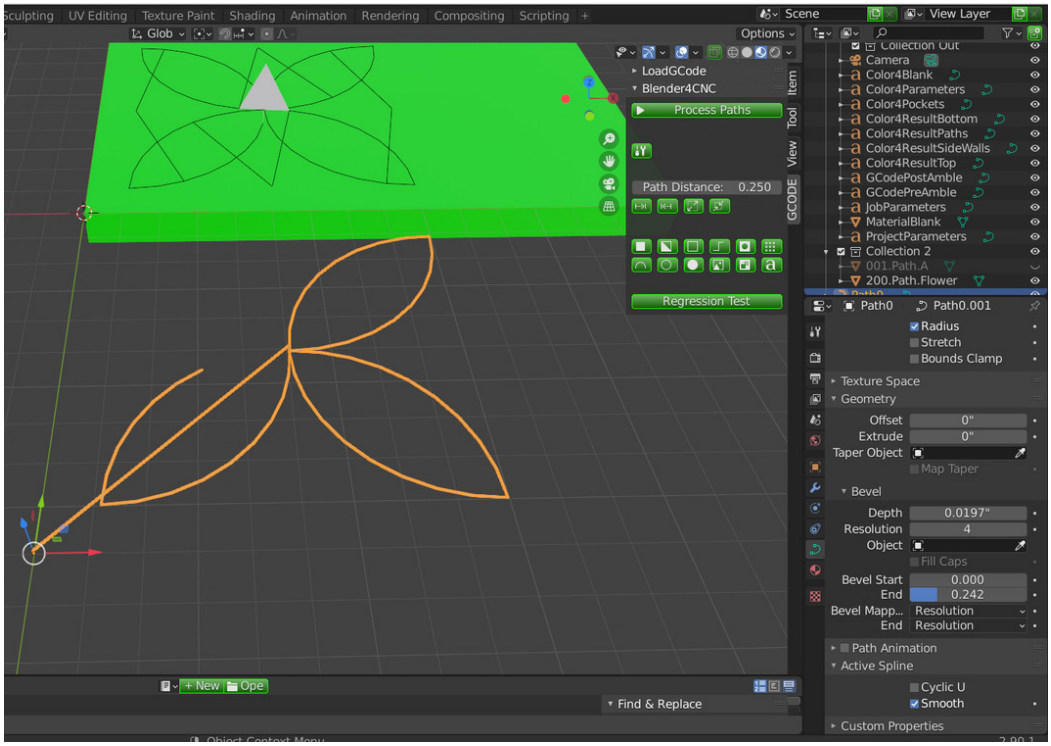

Figure 331: Then the cutter cuts the 3rd petal.

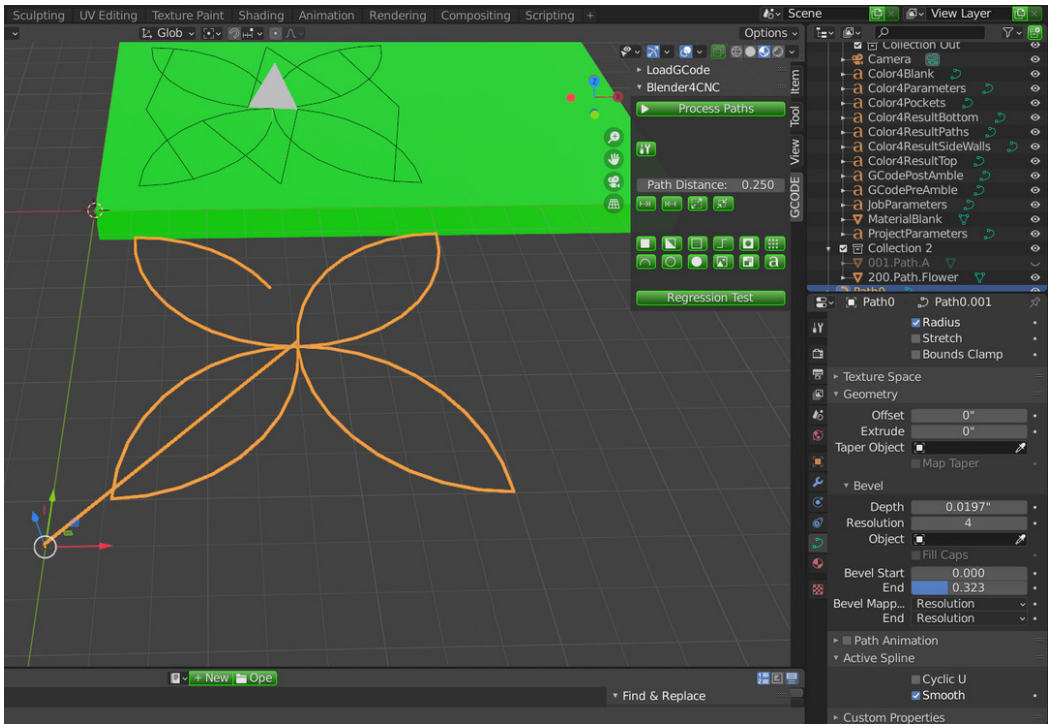

Figure 332: Finally, the 4th petal.

1.11 Working with Drill Paths

1.11.1 Creating a Grill with a Drill Path

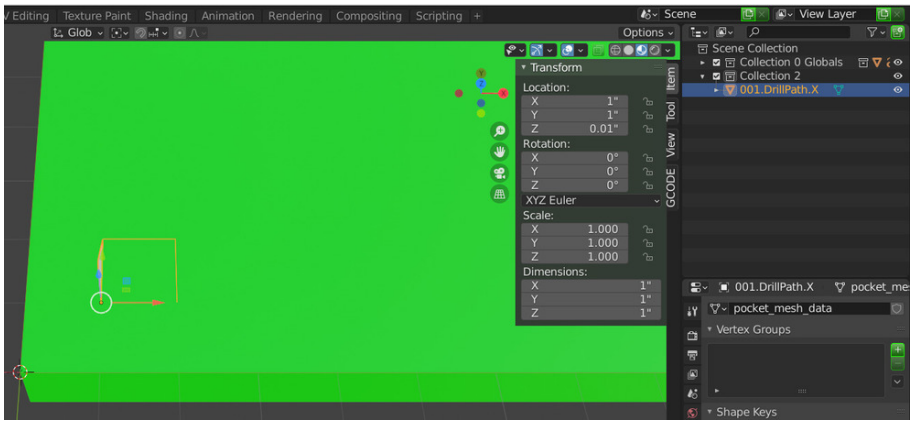

Figure 333: Let's start by adding a drill path. We will be removing the start point, creating a grid of points, and recreating a new start point.

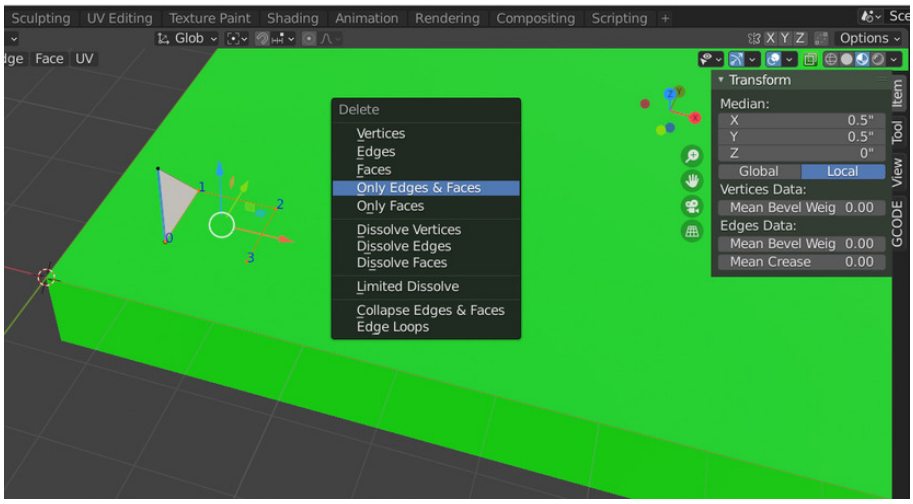

Figure 334: Select the four vertices and press "Delete" and select "Only Edges and Faces".

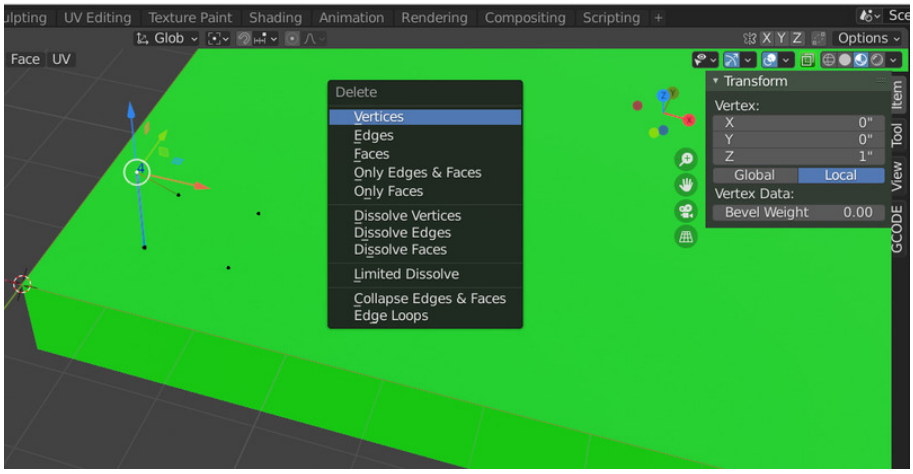

Figure 335: Most of the edges are gone. Select the top of the start point and press "Delete" and select "Vertices".

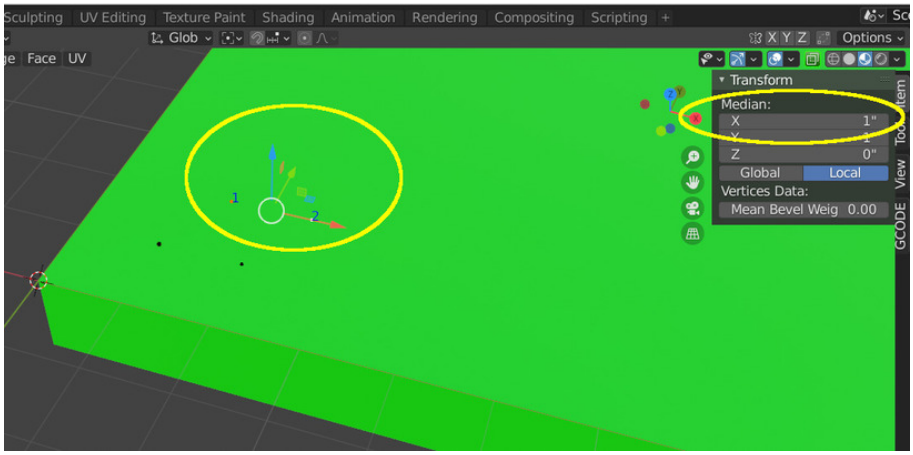

Figure 336: Select the top two vertices (notice how the "3-arrow" axis is now halfway between them and on the right hand side of the display it shows "Median"). Move the median over by an extra 0.5 by typing 1.0 into the Median "X" field. This will offset these points in relation to the row below.

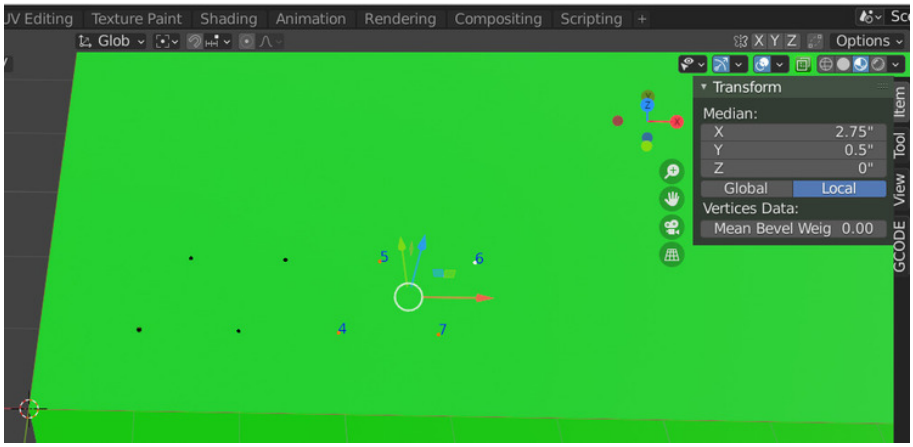

Figure 337: Select all 4 points then press shift-d then press enter. This will copy the 4 points and now you can drag the copied 4 points to the right (holding the ctrl key while dragging the red arrow will cause the vertices to "snap" to the blender grid very conveniently).

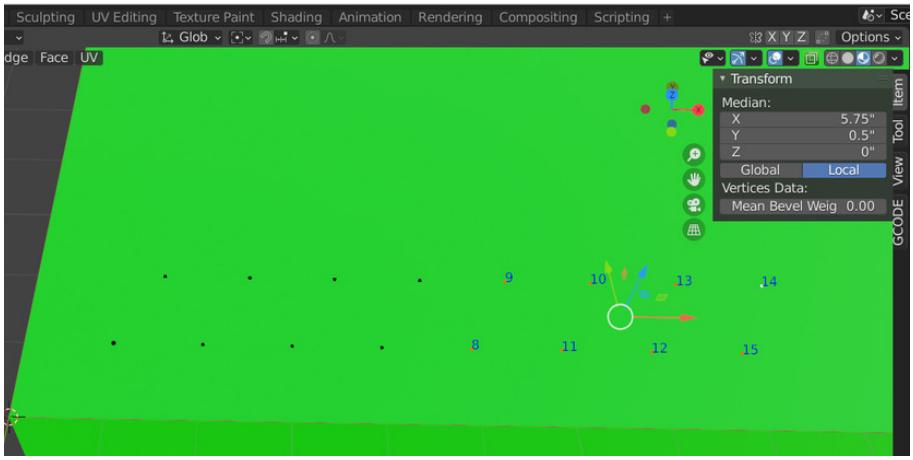

Figure 338: Select all 8 points now and repeat the copy to create 16 points.

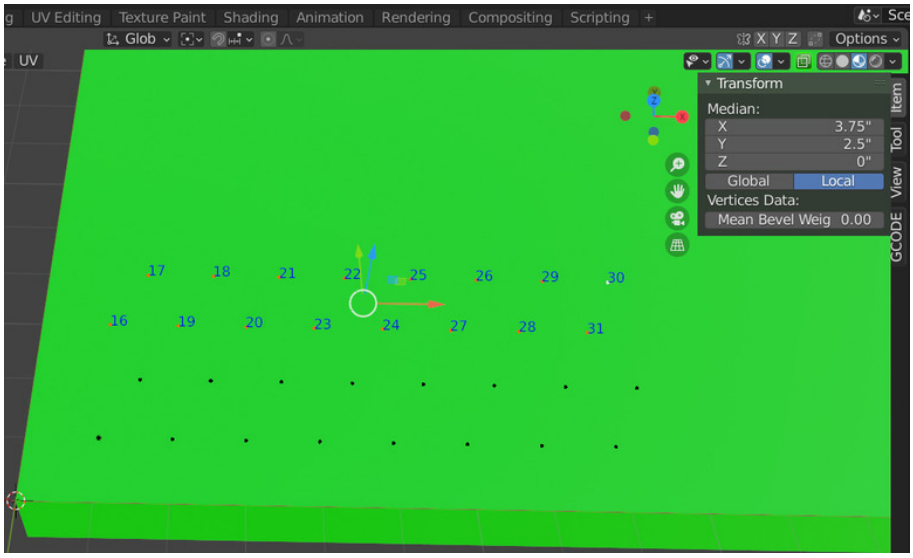

Figure 339: Select all 16 points now and repeat the copy but this time drag the copied points up.

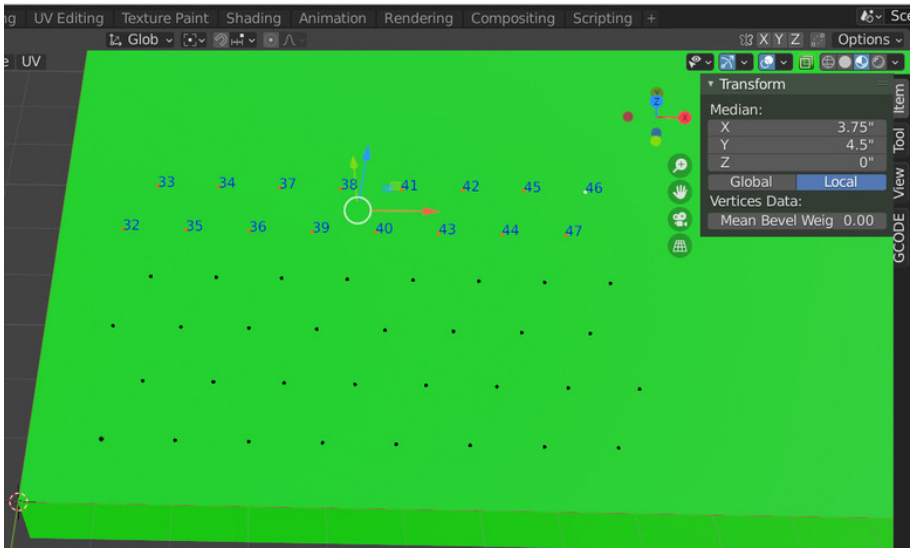

Figure 340: With the top two rows selected (16 points) copy them and move them up so we now have a total of 6 rows of points with every other row offset.

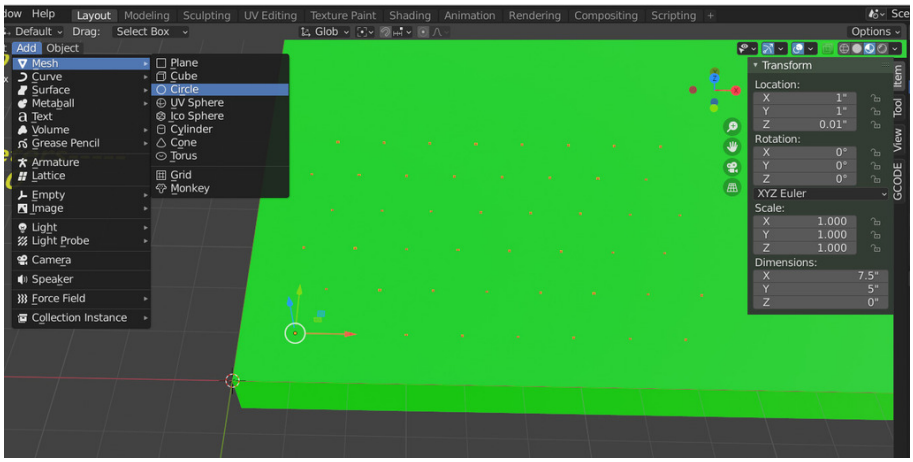

Figure 341: Press TAB to exit mode. Temporarily, add a circle to use as a guide for the next step.

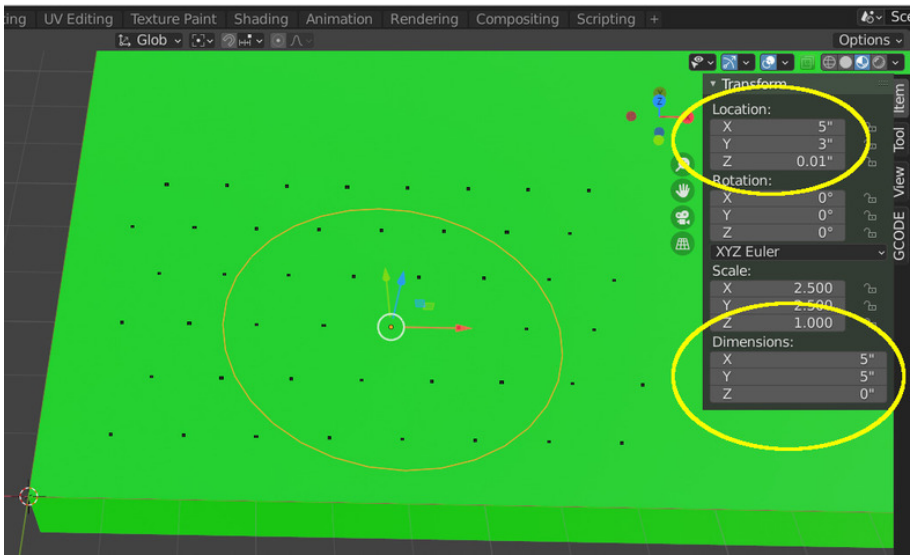

Figure 342: Move the circle to (5,3,0.01) and change its dimensions to (5,5,0). Note the Z location was changed to 0.01 so that the circle shows up more clearly above the top of the material.

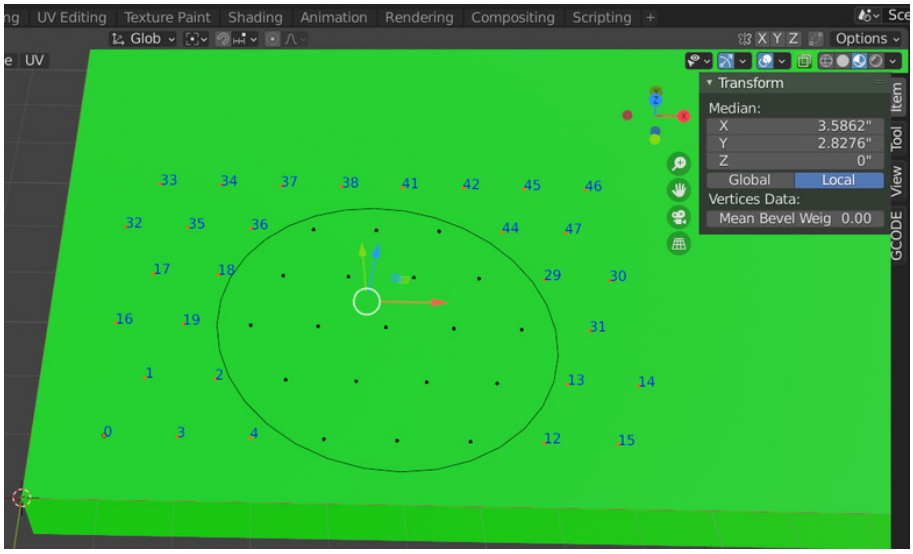

Figure 343: Select the Drill Path object and enter edit mode by pressing TAB. Then start selecting all the points that are outside the circle and deleting them.

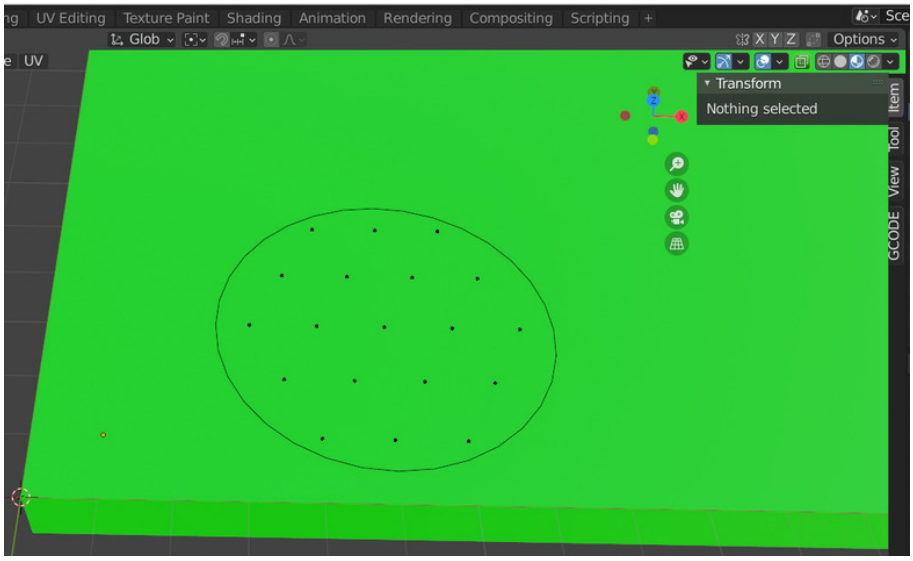

Figure 344: All the points outside the circle are deleted.

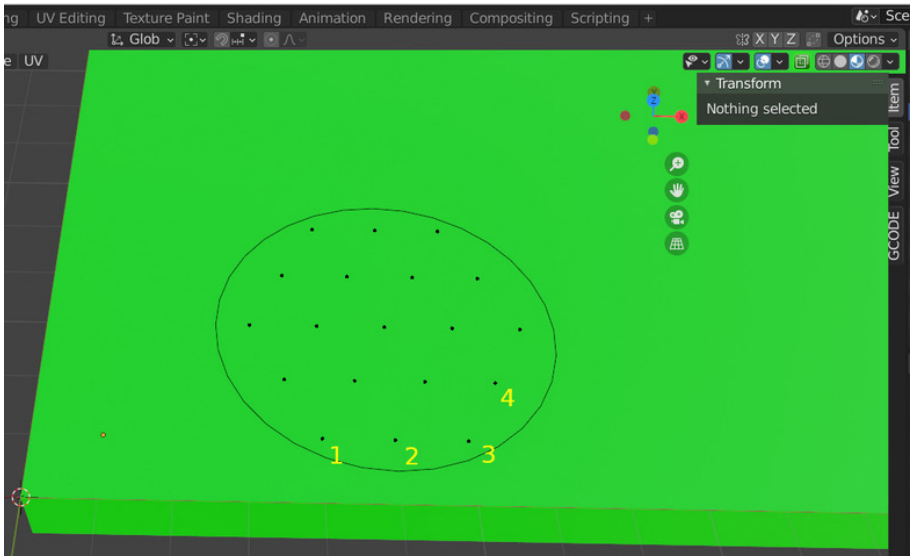

Figure 345: We will now connect all the points into a continuous path. So that the machine moves in an efficient manner we must select the points in a sensible order. Start by clicking on the vertex in the lower-left corner and then shift-click the points in the order shown.

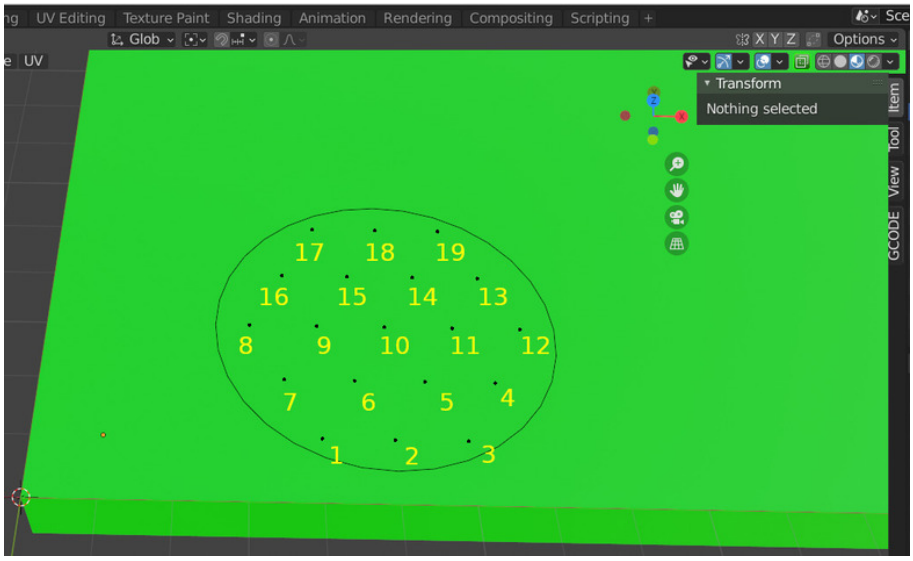

Figure 346: Continue shift-clicking the points in the specific order until they are all selected.

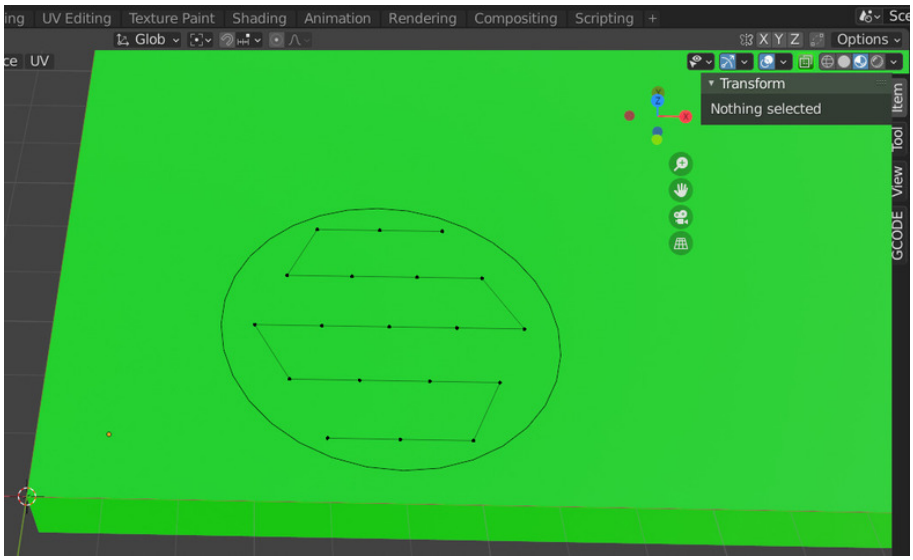

Figure 347: Press "j" to create the edges in the order in which they were selected.

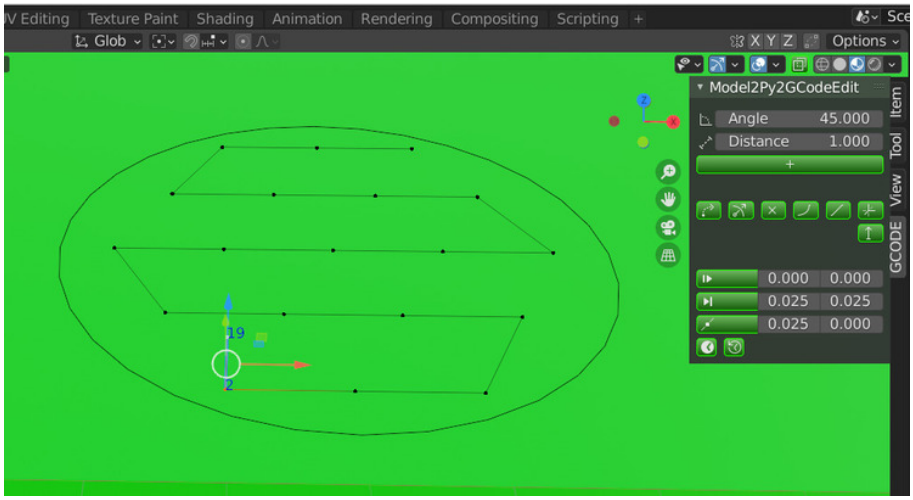

Figure 348: It is now time to recreate the start point. Select the vertex in the lower left and click the "Create Start Point" button.

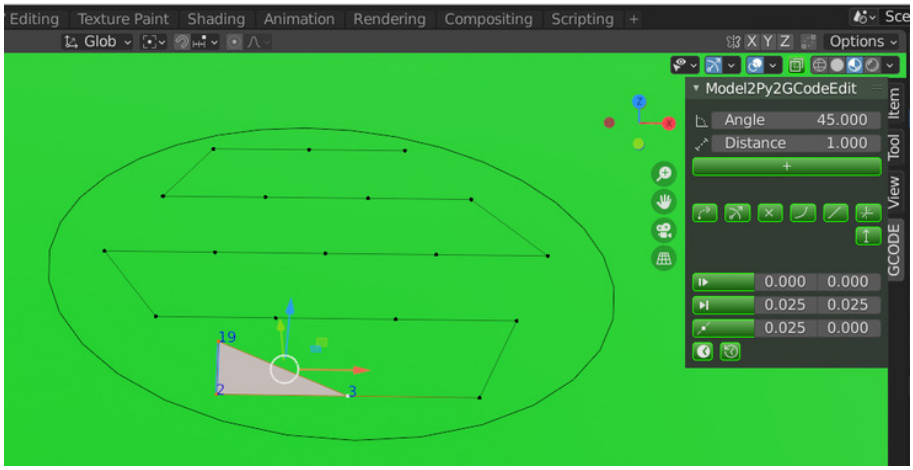

Figure 349: Select the 3 points to form a triangle and press "f" to create the starting direction flag.

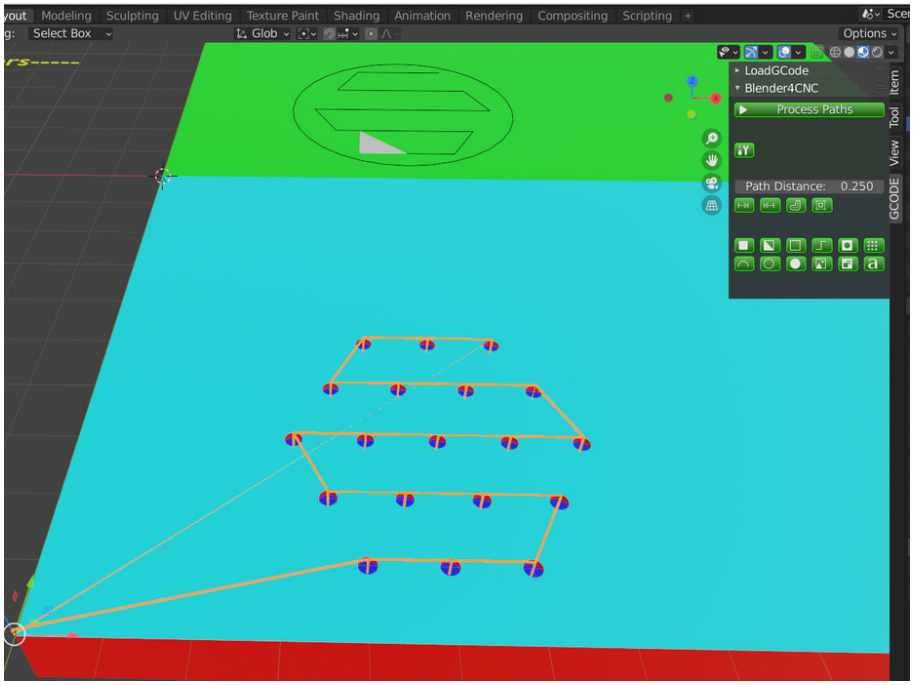

Figure 350: Press TAB to exit edit mode and then click "Process Paths". Notice that the circle object is ignored because it does not match the naming convention of CNC operations. You can place anything in the scene that is helpful and as long as it does not clash with the naming convention it will not interfere with the generation of the GCode. If you don't want the circle anymore you could delete it.

1.11.2 A Circular Drill Path

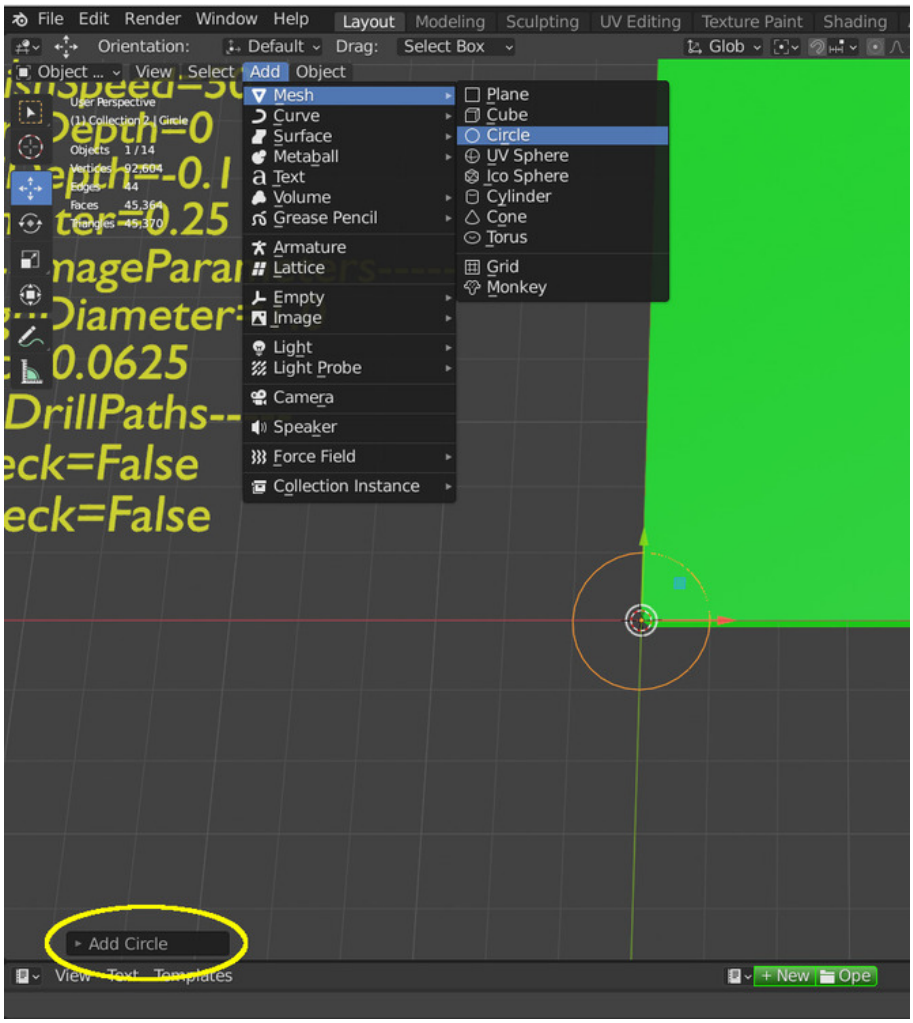

Figure 351: Let's drill a circular pattern of 12 holes using a drill path object. Add a new circle mesh from the "Add" menu. Immediately after doing this you will see a sub-menu at the lower left of the viewport showing the most recent function "Add Circle".

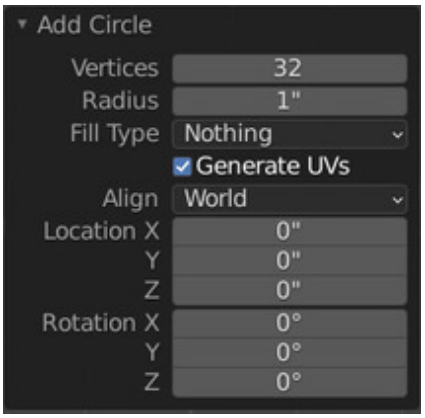

Figure 352: Click the little arrow beside "Add Circle" to open the popup menu.

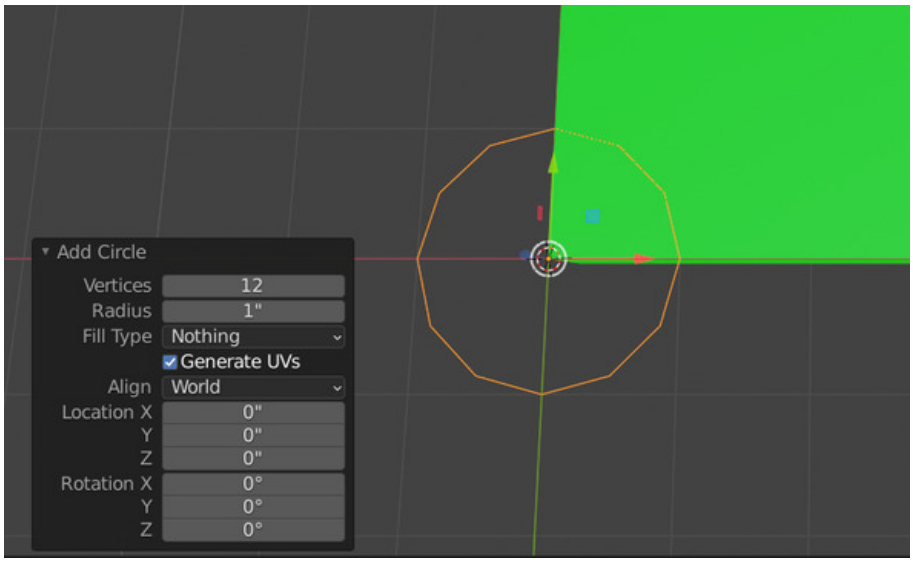

Figure 353: Change the number of vertices to 12 and notice now that the circle that was just added now looks like a 12-sided polygon instead of a smooth circle.

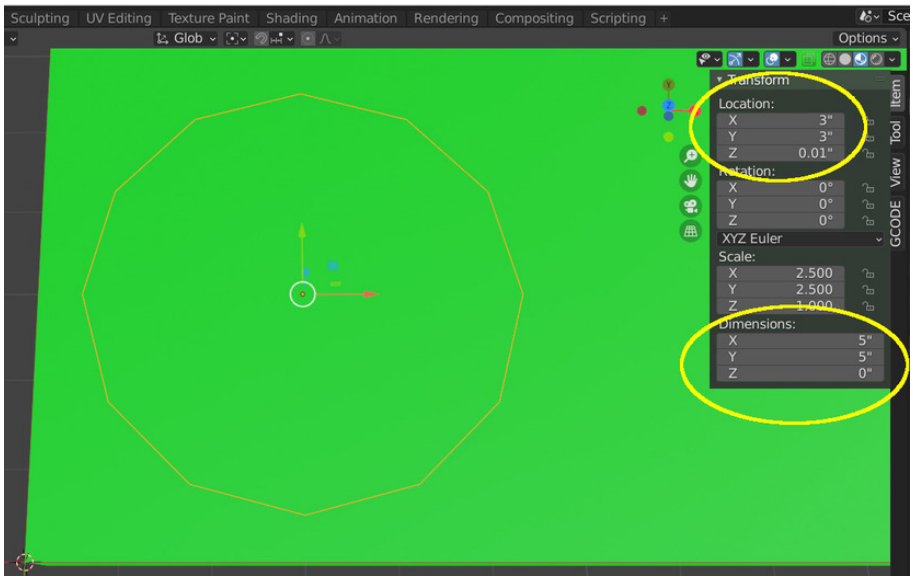

Figure 354: Move and resize the circle.

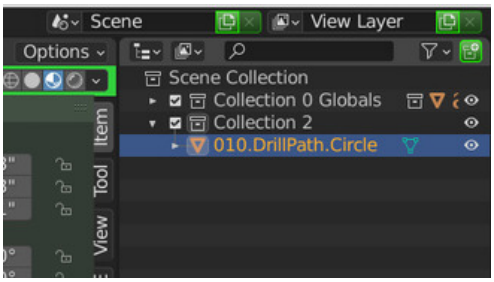

Figure 355: Rename the circle to "010.DrillPath.Circle" and drag the name onto "Collection 2" so that the operation is contained within "Collection 2".

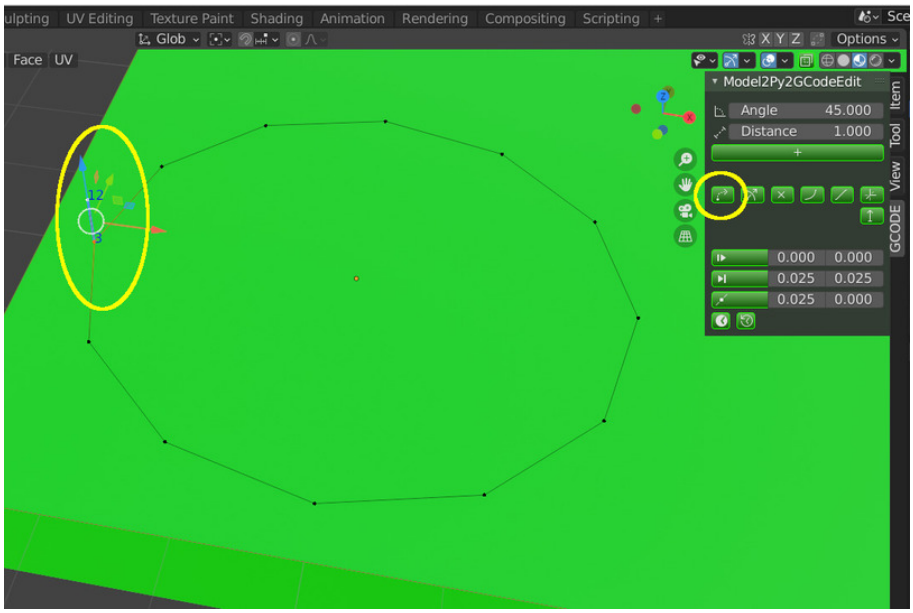

Figure 356: Select the circle and press TAB to enter edit mode. Then select just the left most vertex and click the "Create Start Point" button to make this the start of the path.

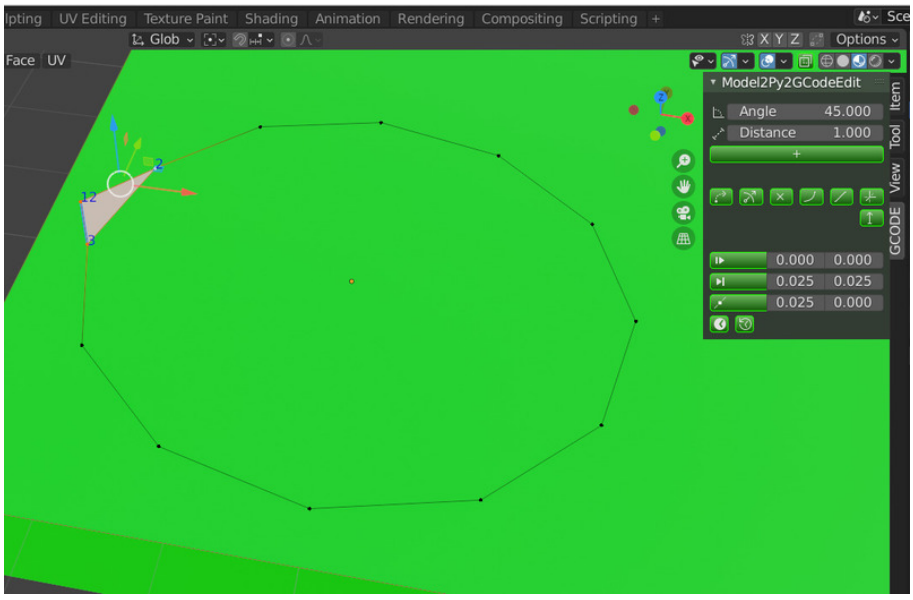

Figure 357: Select the 3 points that make the starting triangle and press "f" to create the direction flag.

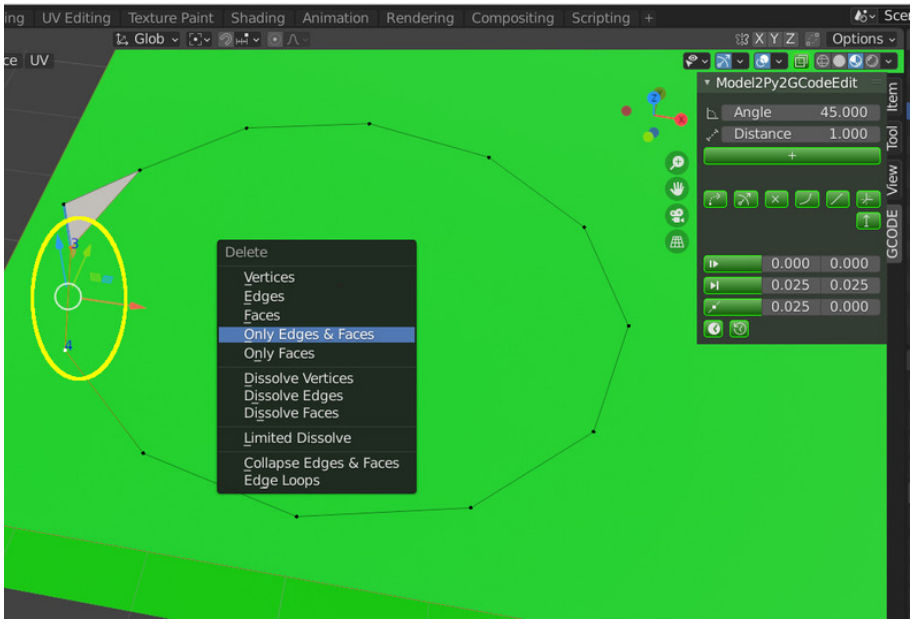

Figure 358: Select the first point and the "last" point in the circle. Press "Delete" and select "Only Edges and Faces" to delete the link from the end of the path to the point we have marked as the start.

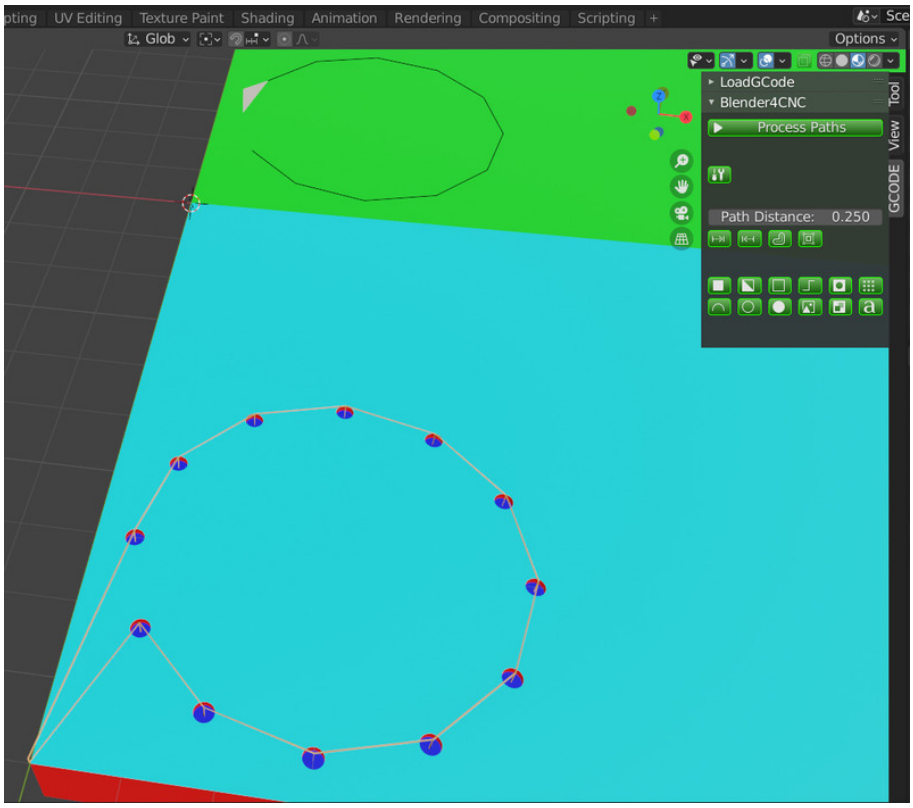

Figure 359: Press TAB to exit edit mode and click "Process Paths". The end result is a circle of 12 holes.

1.11.3 Drill Path Parameters

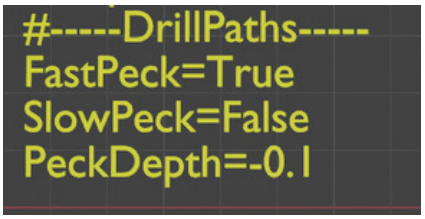

Figure 360: Drill paths make use of three special job parameters "FastPeck", "SlowPeck" and "PeckDepth". If both "FastPeck" and "SlowPeck" are "False" then the holes are simply drilled by having the cutter plunge down and back up in one movement. This is a "G81" drill cycle. Otherwise if "SlowPeck" is set to True then a "G83" drill cycle is generated and this is where the cutter will plunge down a little bit, then fully retract before plunging a little deeper. It will fully retract before it plunges a little deeper until it has reached the full depth. Otherwise if "FastPeck" is True, then it will generate a "G73" drill cycle where the cutter pecks down in short bursts but unlike a "G83" cycle, it will only retract a small amount. The parameter "PeckDepth" sets the amount to "peck". Note that when visualizing a drill path, the curve object generated that represents the path of the cutter will not simulate the pecking action - it will just show the cutter moving down to the hole depth and raising back up (only noticeable if you look very closely at the path during those frames where the cutter is drilling).

1.12 Fixing Meshes

1.12.1 Problematic Points in Meshes

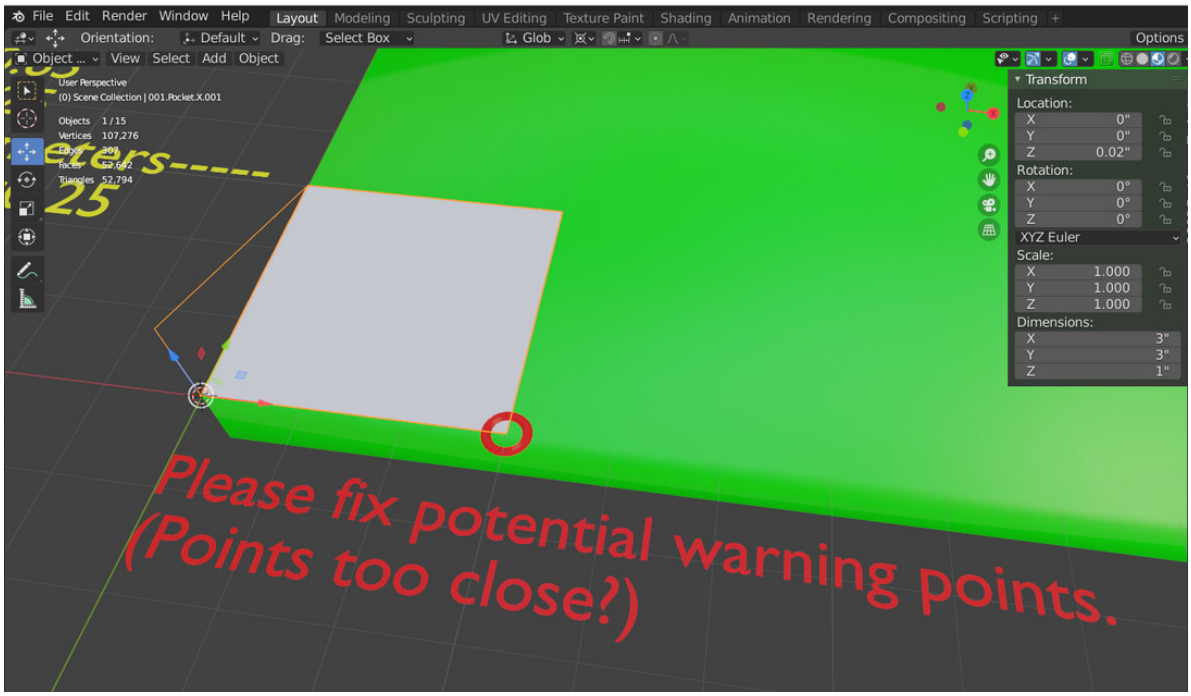

Figure 361: Sometimes, something "weird" can occur with a mesh on an operation. These situations can cause the software to hit unexpected errors due to the precision of the mathematics. Often this is caused by points in a mesh being too close together. This may occur because a user has manually placed points too close OR the complicated shape being processed through functions like Shrink or Expand can result in a mesh that has dangerously close points. (By "close" we mean within about 1/100 of an inch - most woodworking CNC operations do not need mesh points this close when using cutters that are in a range of 1/16" to 1". Even when using very small cutters, you rarely need points along a path to be this close (except for PCBs maybe?)). Here is a case where we tried to process a simple square pocket. The software circled a vertex on the mesh and produced an error message. It thinks there is something about this point in the mesh which it cannot handle reliably.

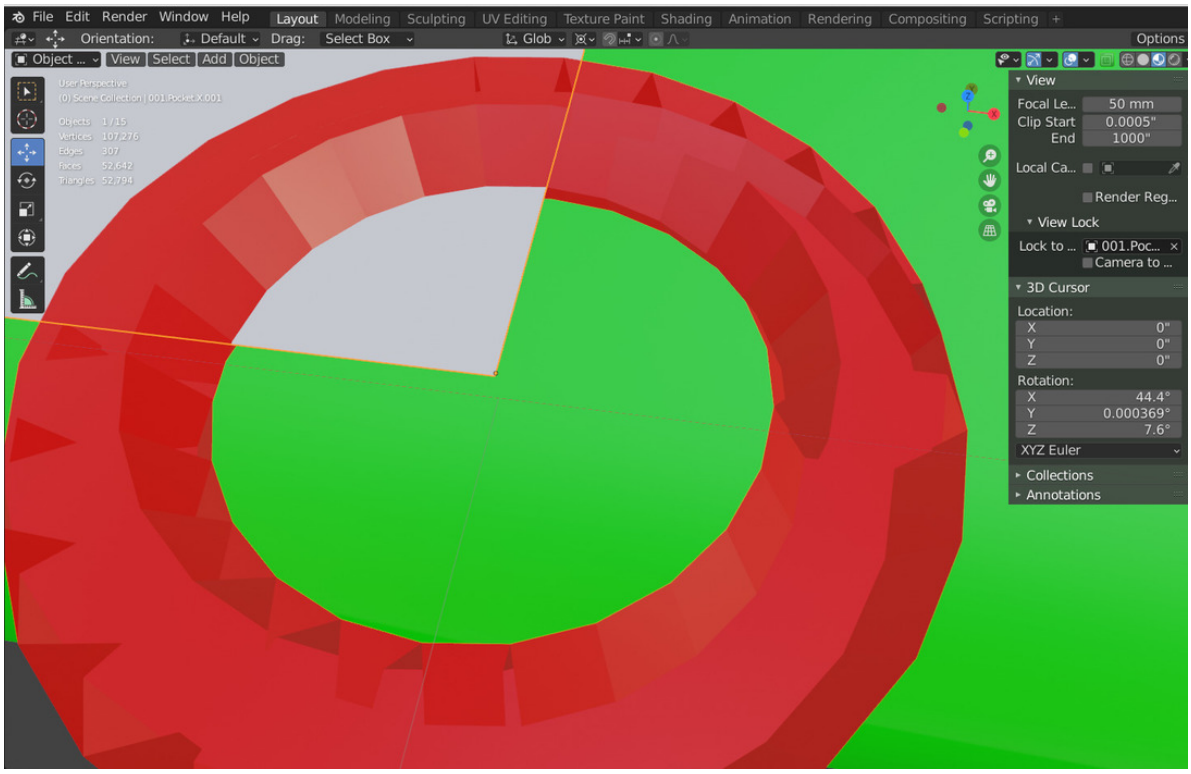

Figure 362: Let's zoom in on the area in question.

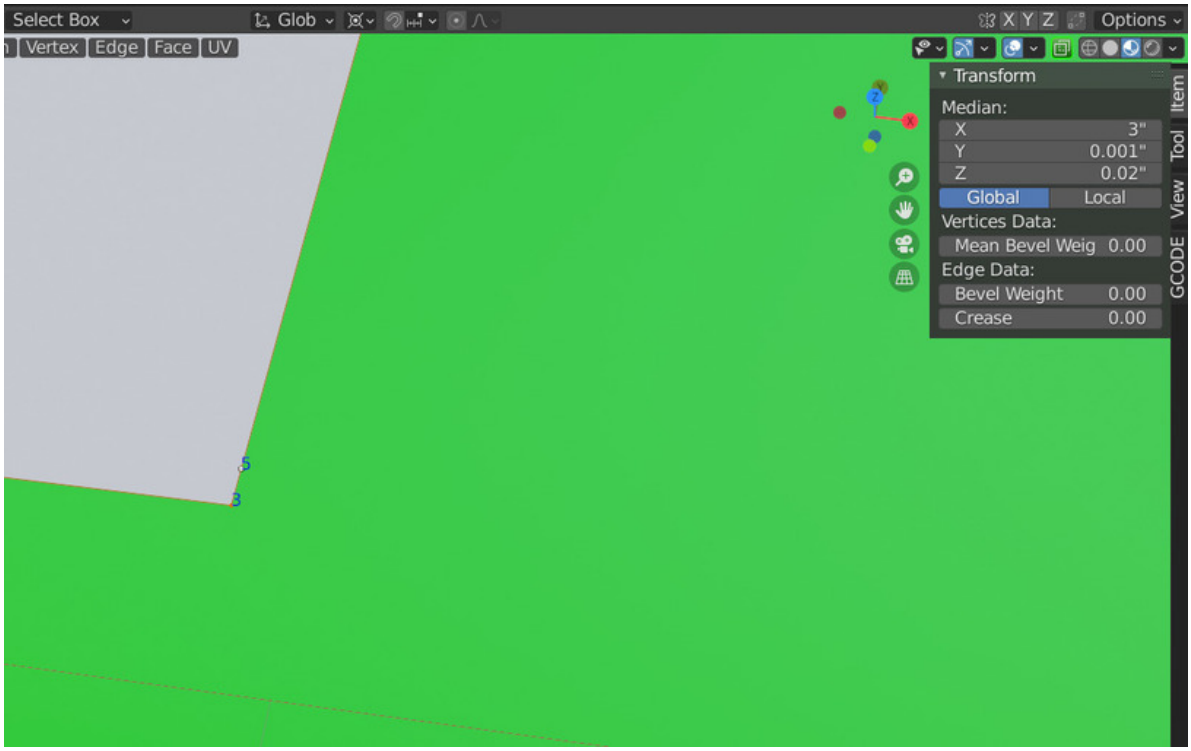

Figure 363: Let's enter EDIT mode and zoom in further. We can see there are actually two points really close together on this corner. This is likely what the software is complaining about.

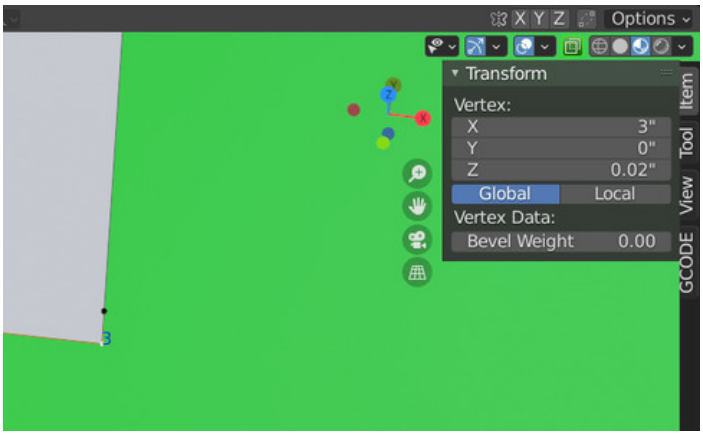

Figure 364: One point is at Y=0.

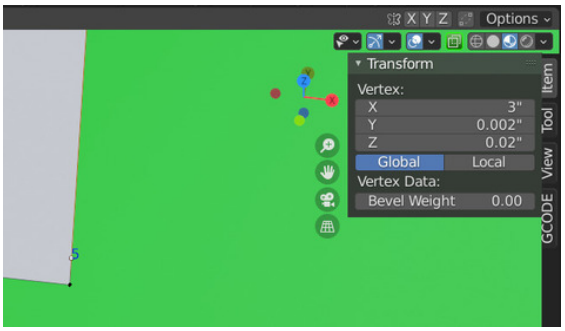

Figure 365: The second point is at Y=0.002

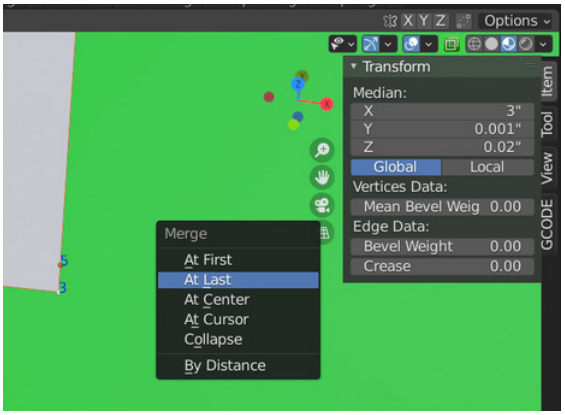

Figure 366: The simple solution here is to delete the "5" point. We could either delete it and then recreate the right-hand edge of the square pocket, or we can select the "5" point, then shift-click the "3" point and press "M" to merge these two points together at the location of the "last" selected point (the "3" point). Merging vertices together in blender does not remove the attached edges and faces.

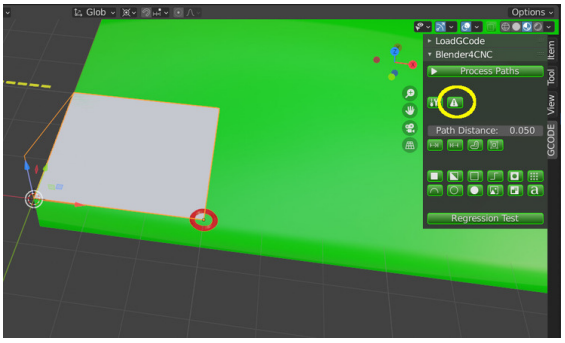

Figure 367: You can check a mesh for problems by clicking the "CheckMesh" button. This function will draw a red circle around any vertices that it thinks will cause a problem. Just select one or more operations and then click the "CheckMesh" button.

1.12.2 Zooming in Blender

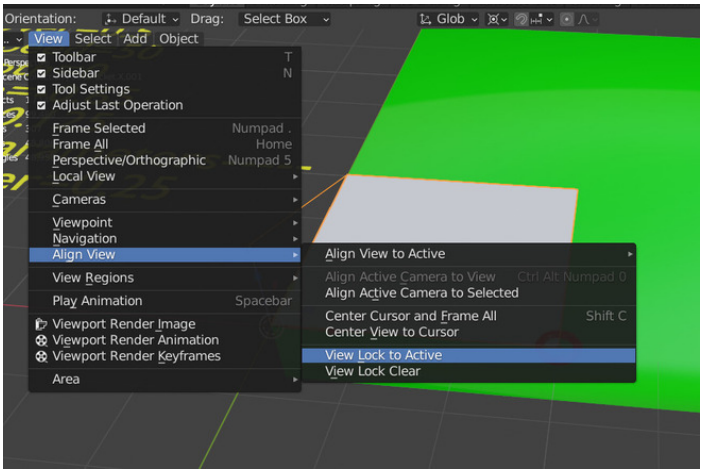

Figure 368: Zooming into very small details in Blender can sometimes be painful. So how to do it without the object of interest disappearing off the edge of the screen? Select the object/operation and go to the "View" menu, "Align View" sub-menu and select "View Lock to Active". Now it is much easier to keep the selected object on the screen while zooming in and out.

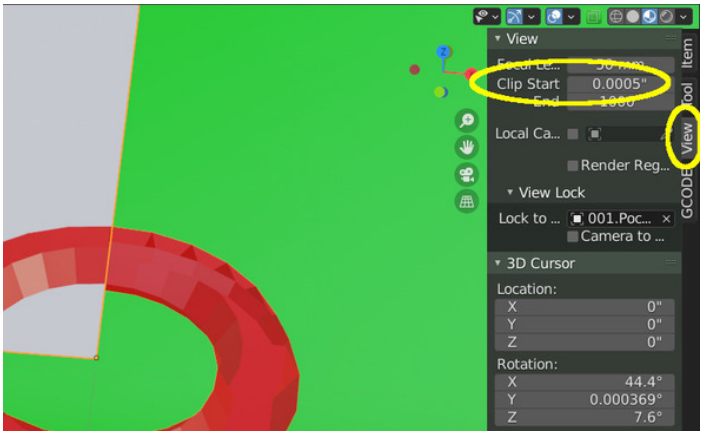

Figure 369: Sometimes when you zoom in too far everything disappears! This is most likely due to the "near clipping plane". (In programs like Blender, things that are **really** close to your point of view are deliberately made invisible.) Go into the "View" tab menu and set the "Clip Start" to a really low value. Now you can get much closer to your object before the clipping will occur.

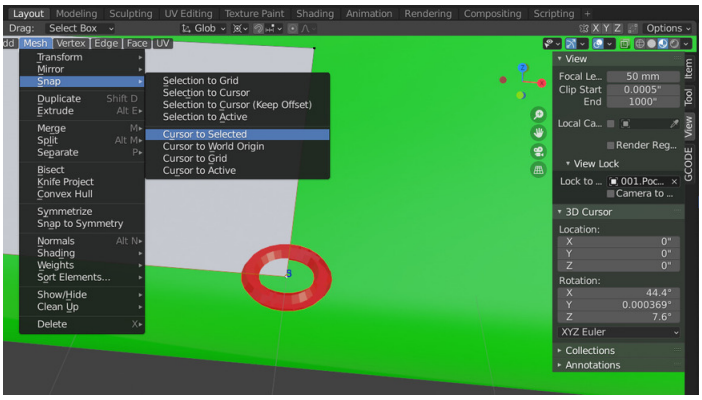

Figure 370: When the view is locked to an object, zooming in is also affected by the "origin" of the object. All the vertices within an object are relative to a origin point (which may or may not be on one of the object vertices). Often the origin of an object is somewhere to the lower left or maybe in the center. But if you want to zoom in to an area of an object that is far away from its "origin" you will have a lot more success if you temporarily move the object origin to your point of interest. In Edit Mode, select the point of interest (in our case here we have selected the point in the middle of the error circle (actually there are two points there and we don't really know if we have selected the one at the actual corner or the very close erroneous point but either one will let us zoom in to the area to take a closer look). Then go to "Mesh", "Snap" and "Cursor To Selected".

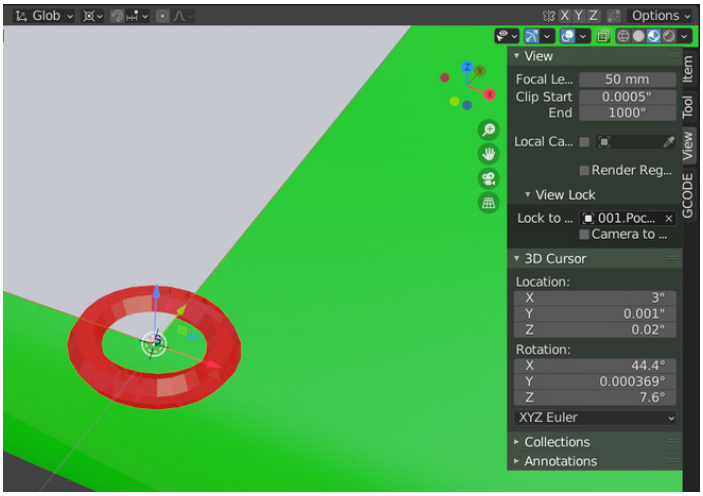

Figure 371: The immediate region looks a little more cluttered as Blender's 3D cursor has been positioned right on top of our selected point.

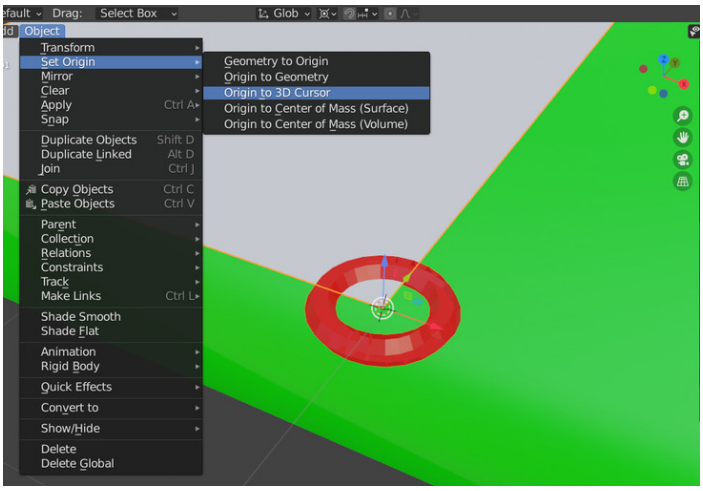

Figure 372: Exit Edit Mode and go to "Object", "Set Origin", then "Origin to 3D Cursor". Now, all the vertices are relative to this object origin and as we zoom in, Blender will keep the origin in view and therefore all nearby points will remain in the field of view much more easily. You can now enter back into Edit Mode and zoom way in and see the points clearly.

1.12.3 Automatically Fix Problematic Meshes

Figure 373: The "Clean Meshes" button can be used to fix some problems with meshes having points that are too close. It will merge points that are deemed too close to each other. It **may** not do exactly what a user would prefer; it is assumed that any imperfections introduced are so small that they will not be noticed in the finished product.

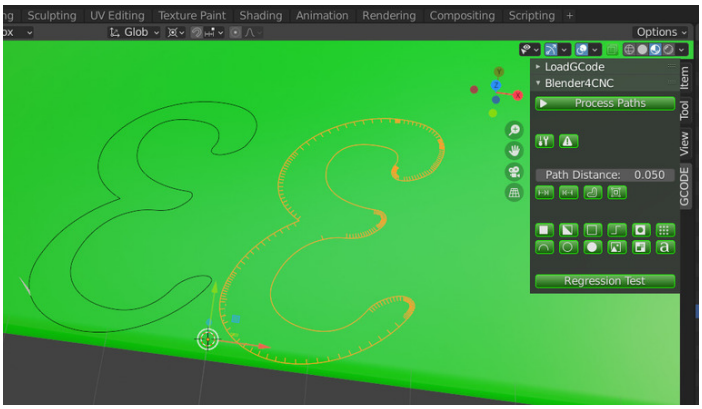

Figure 374: Here is a path that was generated by expanding the cursive "E" shape.

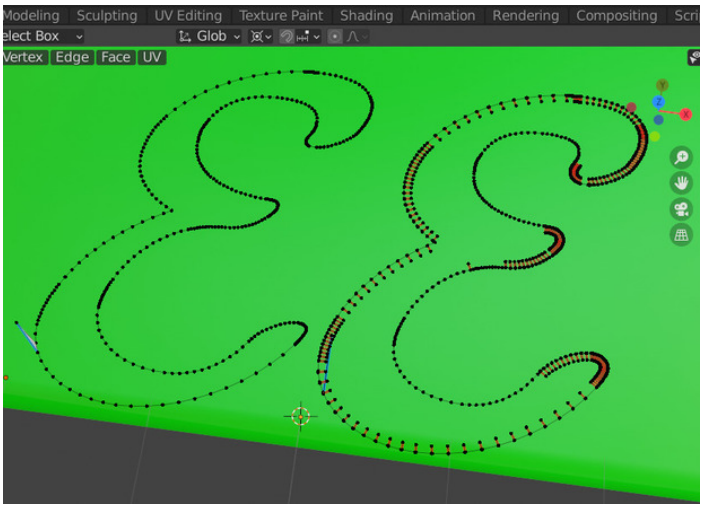

Figure 375: The original operation has a lot of points (probably unnecessarily so). Once expanded, the new operation has many tiny curves that were generated around many of the points. (This occurred in part because the expansion distance was only 0.05"; a larger distance would have chanced that points would have been merged during the expansion. But as it stands, each little straight line bend in the original shape added a tiny curve.)

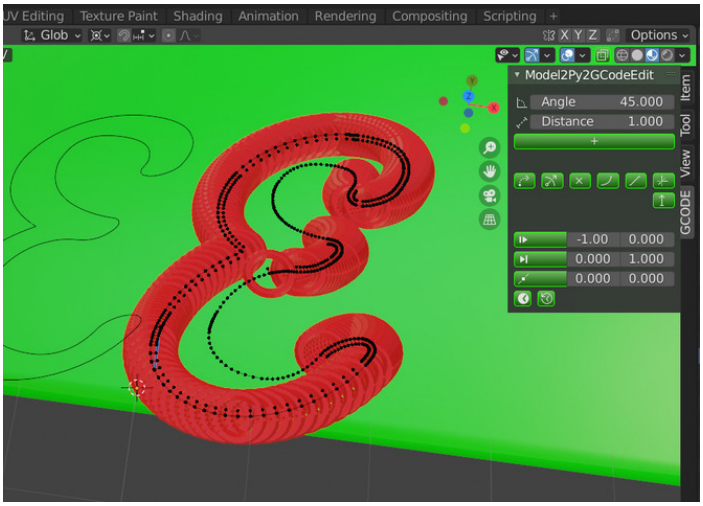

Figure 376: Select the expanded path and clicking "Check Meshes" shows us just how many "problems" exist in the mesh! If there were only one or two problematic areas, a user would be willing to manually correct them but this is a **lot** of points to fix. Each curve is so tiny that the "end point" is **extremely** close to the curve "start point". These curves are so tiny that it will be quite acceptable to just turn each of them into a single point on the path.

Figure 377: Select the operation and click the "Clean Meshes" button.

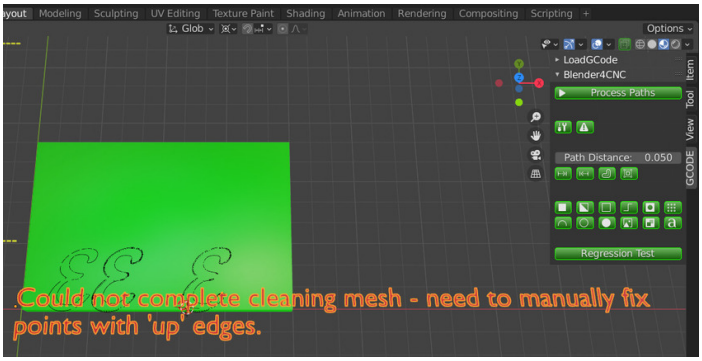

Figure 378: Unfortunately the "Clean Meshes" function will not handle problems that occur around the start flag of an operation. These must be handled manually. Let's zoom in and fix this problem manually and then we will run the "Clean Meshes" function again.

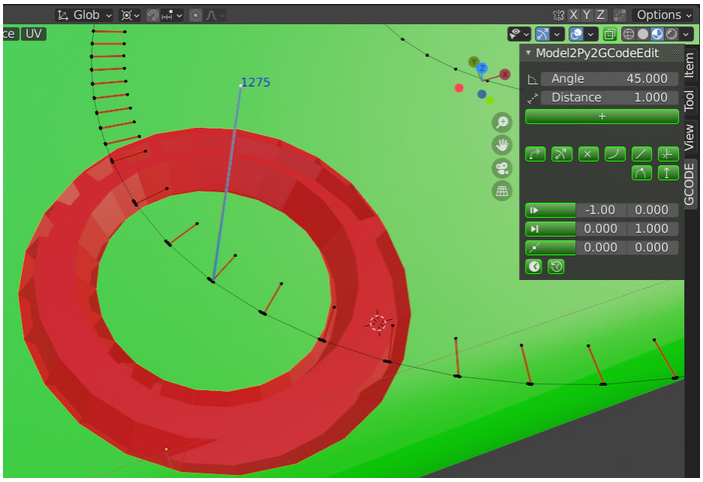

Figure 379: Zooming in, we can see that it has circled the start flag. A tiny curve is part of the start flag. Let's get rid of the few nearby tiny curves and create a start flag that is not on a curve.

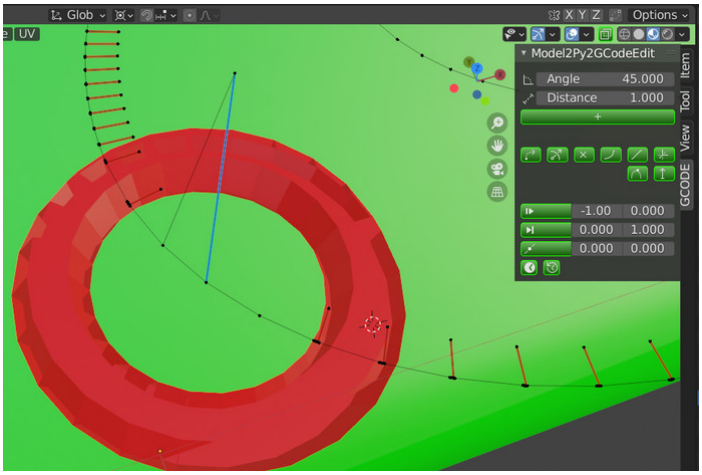

Figure 380: This looks better. No tiny curves are close to the new start flag.

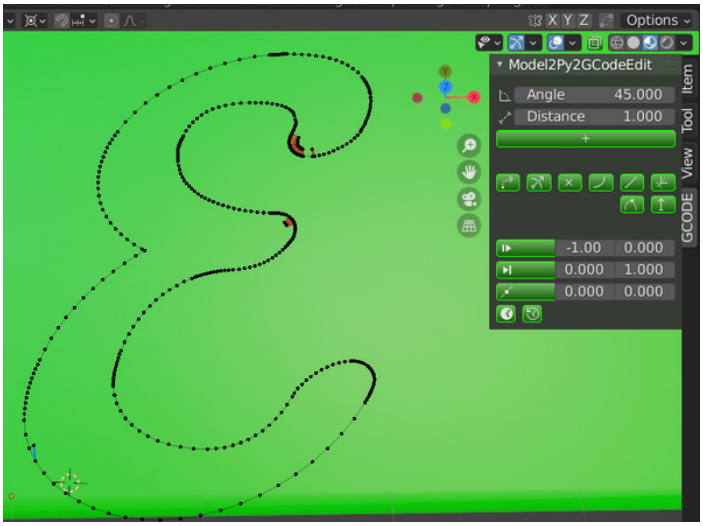

Figure 381: Select the operation again and click the "Clean Meshes" button. The end result is that most of the tiny curves have been eliminated. The remaining curves must be large enough that they do not trigger the checks that the software does for points that will interfere with the math precision.

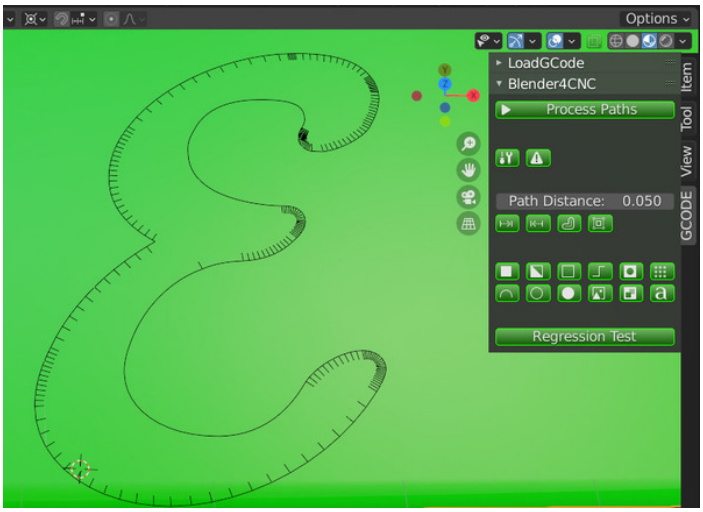

Figure 382: You may wish to clean up a mesh by removing a complex area of tiny curves in just one region of the shape (regardless of whether they are actually "problematic" points or not). Here is the original shape again with all the tiny curves included. Let's remove just one region of curves. We will use the "Reduce curves to a single point" function in edit mode.

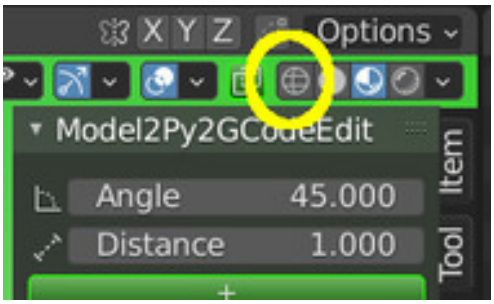

Figure 383: Change the 3D view to "wireframe" by clicking the wireframe button. This is necessary because of the need to make sure that **all** points in a region do truly get selected. Points that are very close to each other can "hide" behind each other when the 3D view is in any other mode than wireframe. This means that you can select a region of points (and they all **look** selected) but in fact some points in the region are **not** selected because they are considered "behind" other points. In wireframe view, ALL points will get selected when you box highlight and select a region.

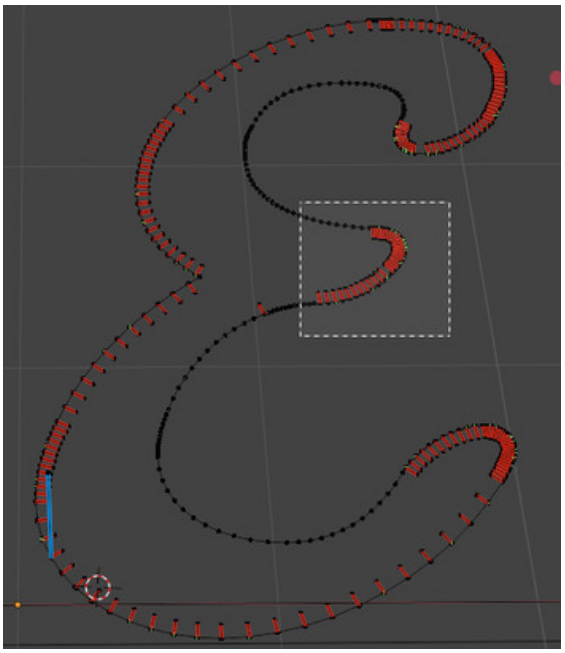

Figure 384: Box highlight an area to select just those vertices.

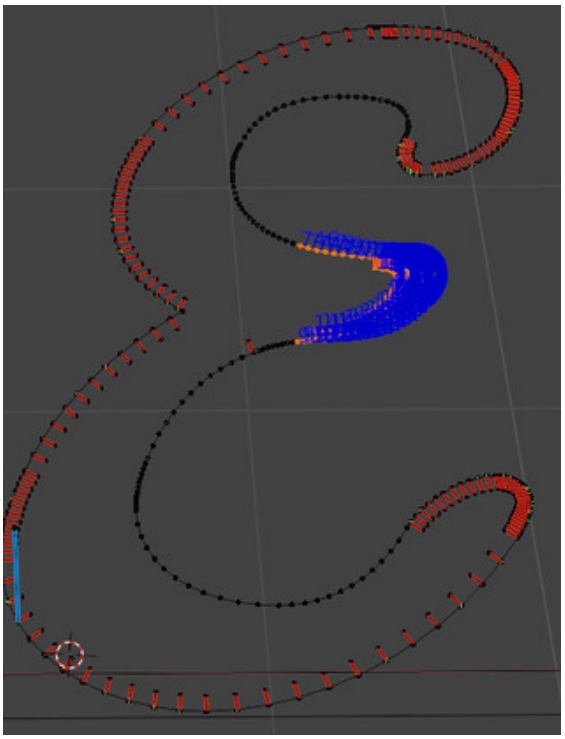

Figure 385: Highlighted vertices are selected.

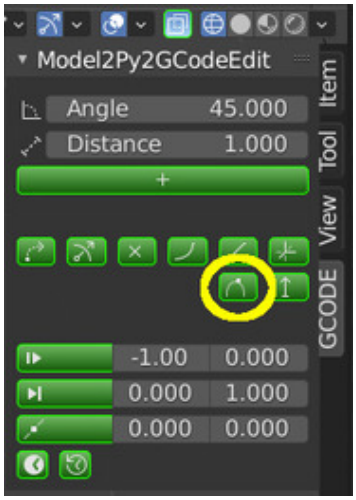

Figure 386: Click the "Reduce curves to a single point" button.

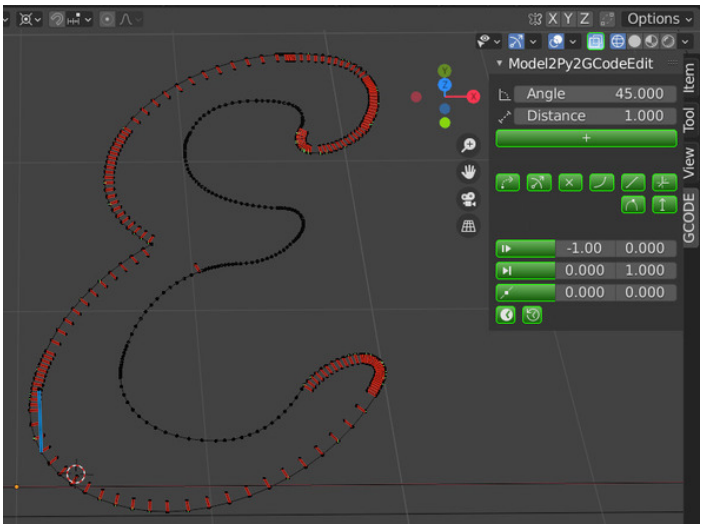

Figure 387: All those tiny (and problematic) curves in the selected region have been removed and each curve was replaced with a simple point so the overall shape of the operation has been retained.

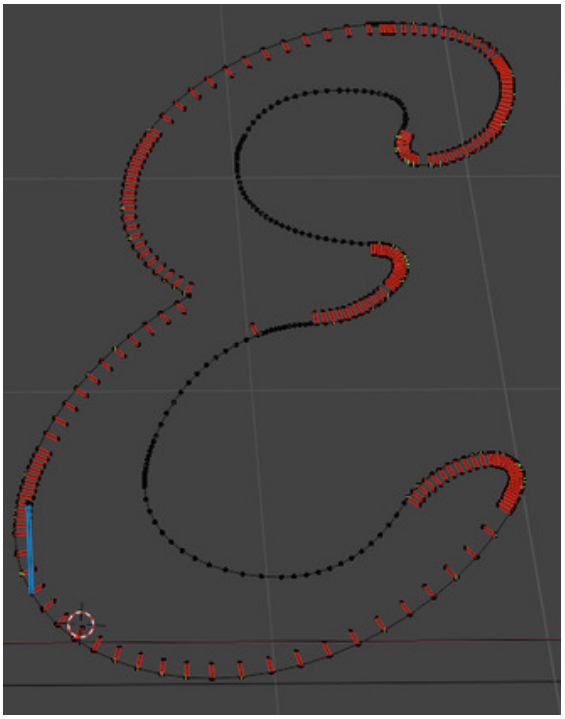

Figure 388: Here is the original shape again with all its original tiny curves. Let's use the same function to remove ALL of them. Note that we are in wireframe mode.

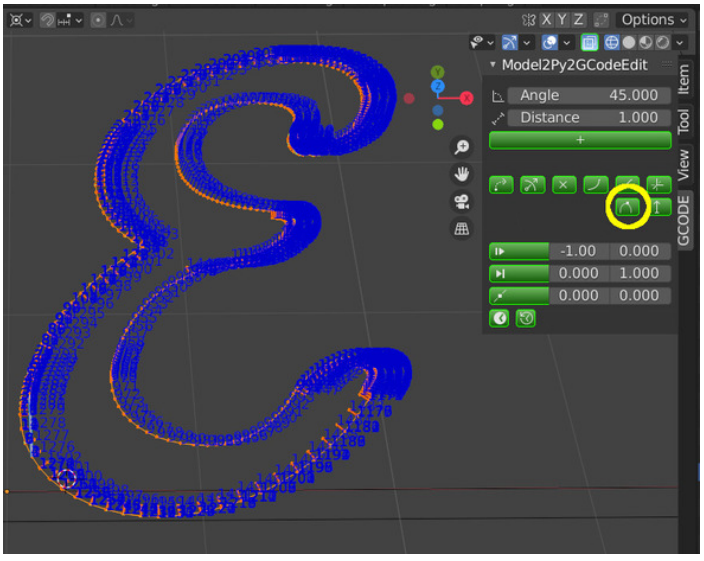

Figure 389: Select ALL points (box highlight or press "a"). Then click the "Reduce curves to a single point" button.

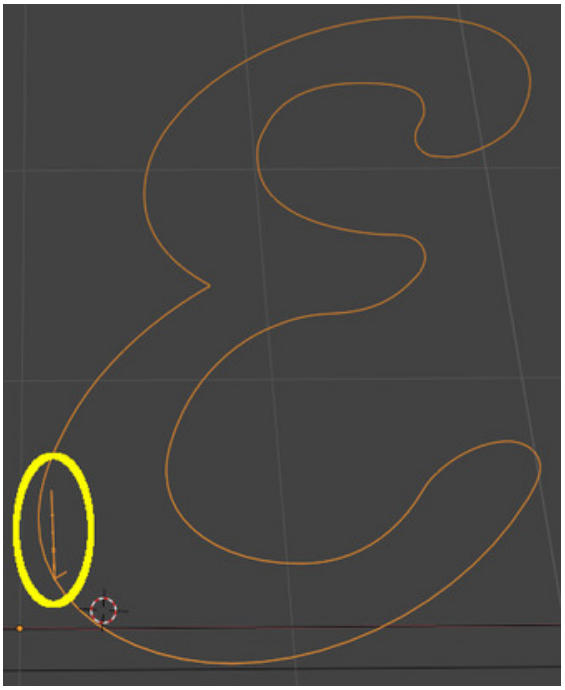

Figure 390: All of the tiny curves have been replaced with a single point on the shape. We will however zoom in closer to the start flag.

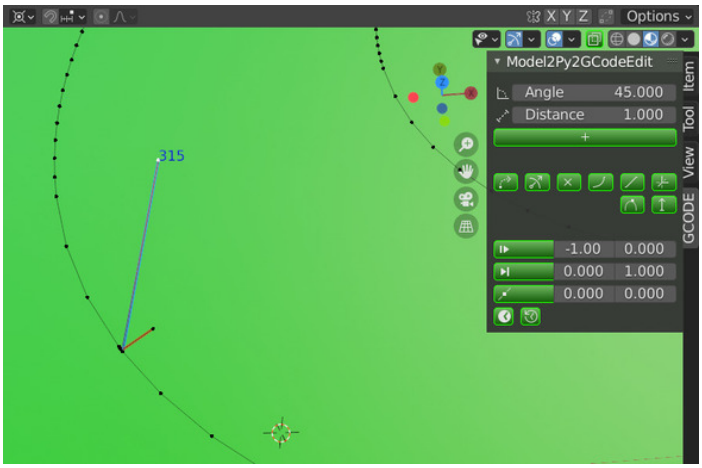

Figure 391: The "Reduce curves to a single point" function will not alter any curves that are ill-formed or part of a start flag. This tiny curve will have to be removed manually if it is a problem.

1.12.4 Example - Expanding Shapes to Trigger Problems

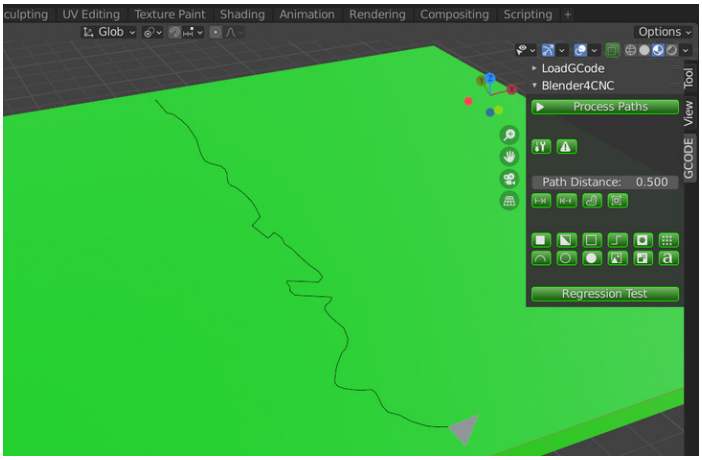

Figure 392: Let's take an unusual shape and use it to demonstrate how we can **cause** problems in a mesh and how we can fix those problems. Here is a squiggly path that looks a bit like a river. We are going to expand it multiple times.

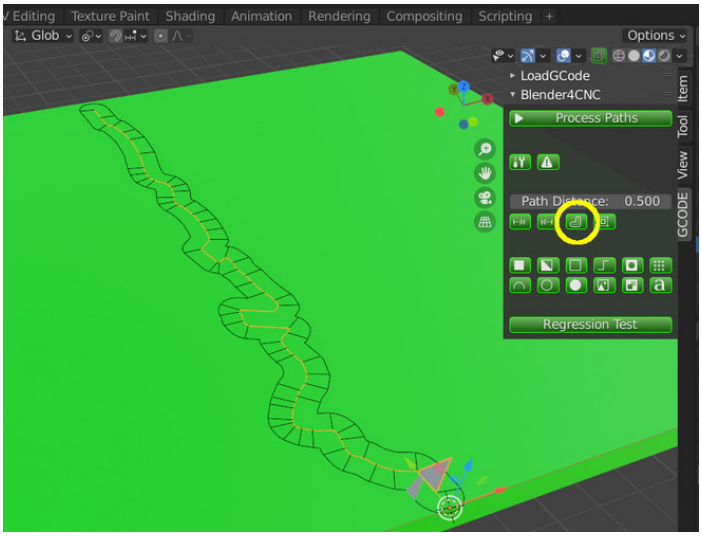

Figure 393: Select this shape and click the "Expand" button to generate a new operation that is 0.5" distance from the original squiggle. An expansion of a shape is generated by simulating how a 0.5" cutter would roll around the original squiggly shape. You can see all those "spines" that go from the original shape to the perimeter: each "external" corner in the squiggly line generated a curve segment as the cutter "rolled" around the corner. Those spines are really the radius lines of curve segments.

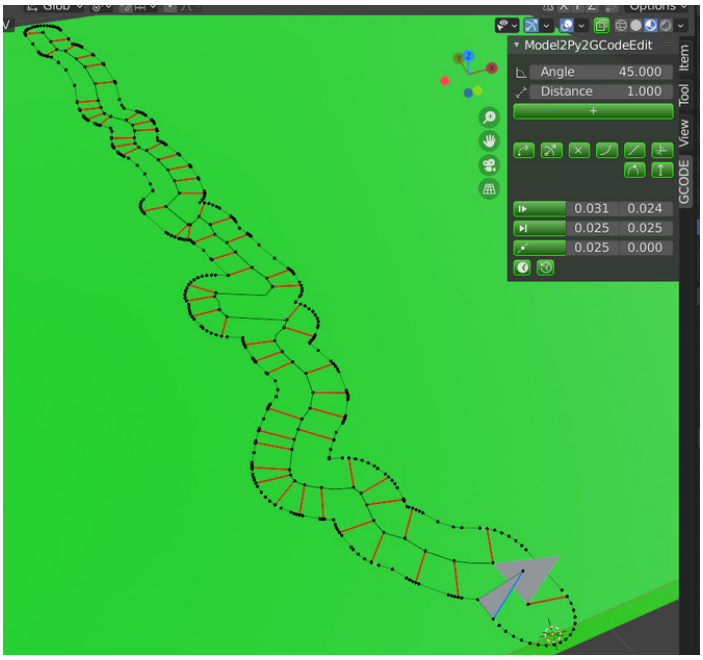

Figure 394: These curve segments are going to cause some problems if we continue. Select the new shape and click Expand again.

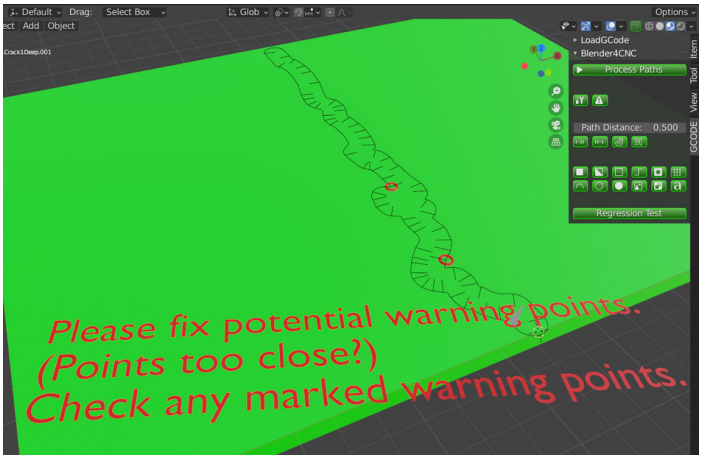

Figure 395: The Expand function did not complete. It has produced a warning that we may have some points that are too close to be processed safely. It has also tried to be helpful and draw red markers around the points of concern.

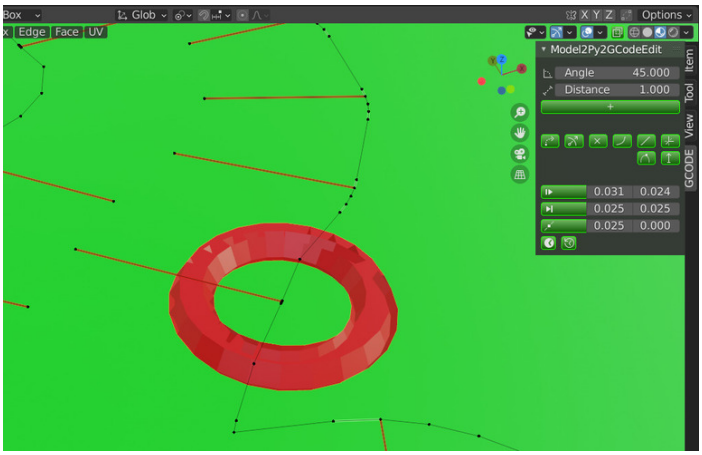

Figure 396: Here, we have entered edit mode and zoomed in on one of the regions of the warnings. These points defining this tiny curve are so close that they are likely to cause math precision errors.

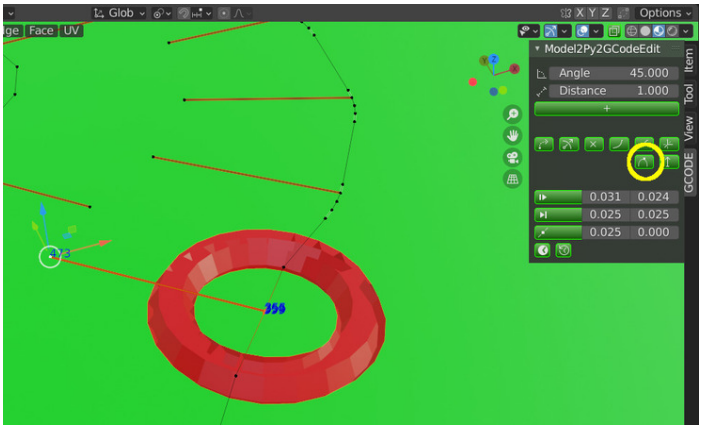

Figure 397: Select all the points of the curve (including the center of the radius) and the click the "Reduce curves to a single point" button.

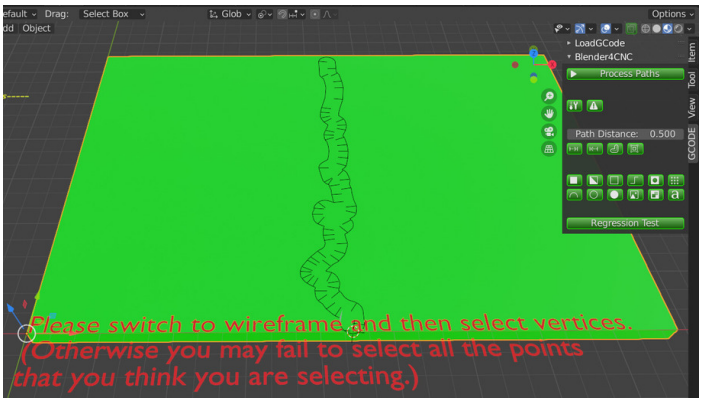

Figure 398: Oops. We failed to be in wireframe mode when we selected the points.

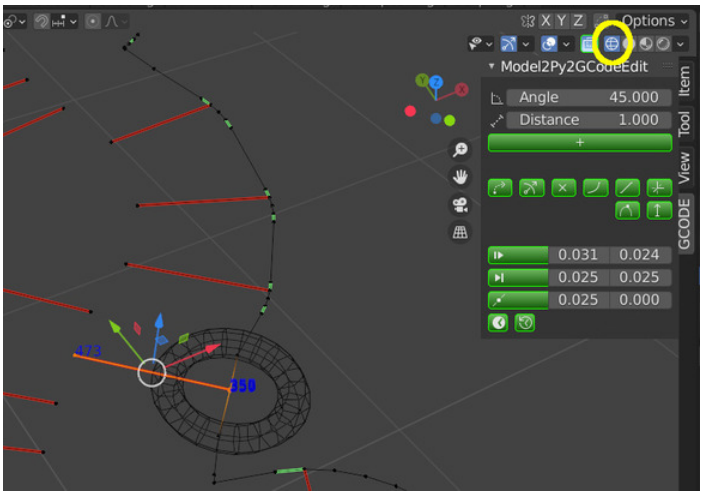

Figure 399: Enter wireframe mode and reselect the points again. Then click the button to reduce the curve to just points.

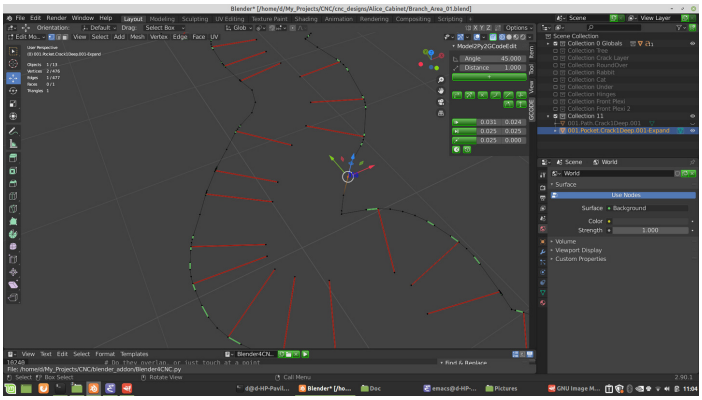

Figure 400: The curve has been eliminated. It has been reduced to just the start and end point with a straight line between. After we reduce any other curves in warning areas, we can run the Expand function again.

Figure 401: What is going on? We eliminated the tiny curves it was complaining about and still the Expand function is warning us about the same areas being problematic.

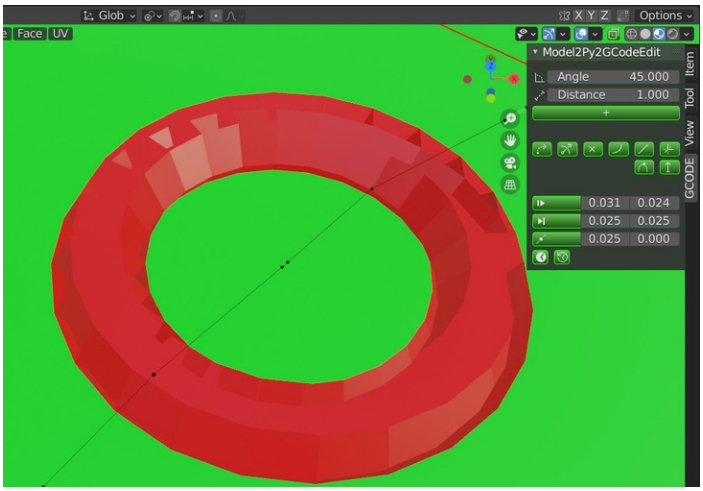

Figure 402: When we zoom way in, we can see that there are actully points in the middle of the warning areas. These are the original start and end points of the curves we eliminated. even though we reduced the curve to just a straight line, the points are still too close together.

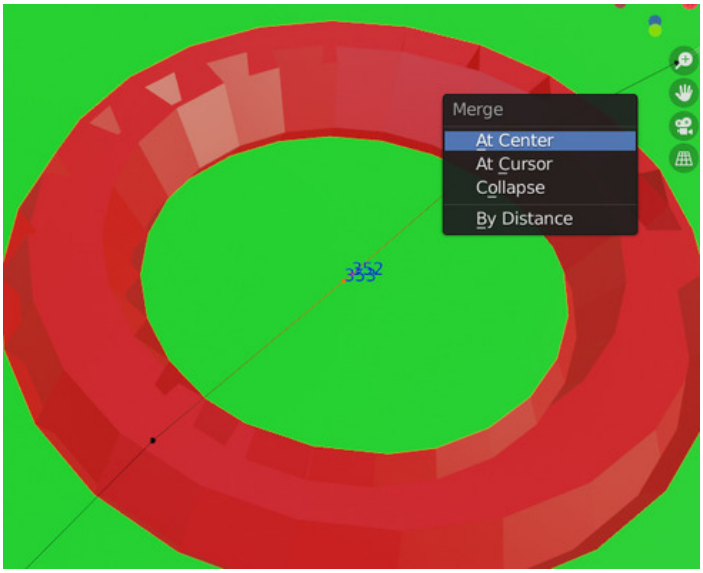

Figure 403: Select both points and merge them (press "m") together at the center. Do this also in any other warnings areas where we eliminated the curves.

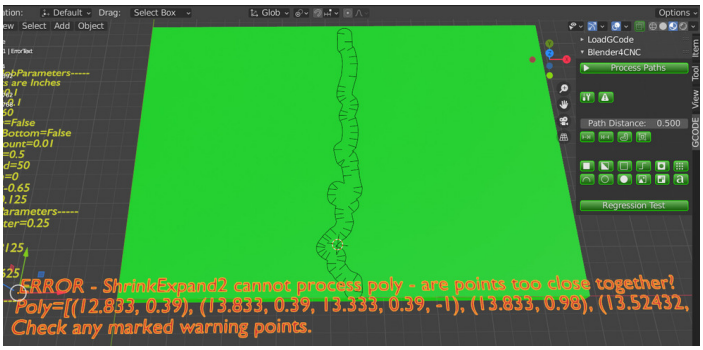

Figure 404: Well after fixing those warnings, we tried to expand the shape again and all we got was another error message and nothing is marked as a warning point in the shape. It turns out that numbers that are "close to zero" are most likely to cause math precision errors so any vertices that are close to the axes (X = 0 or Y = 0) are likely the culprits that have triggered this bug in the software. A trick you can try is to move the whole operation away from the axes and try again.

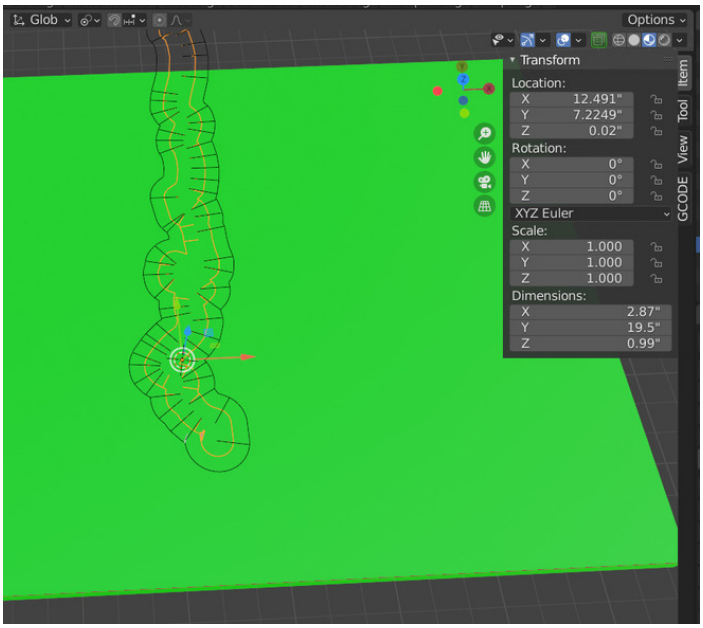

Figure 405: By moving the operation up a few inches, the Expand function now succeeds and has produced a new expanded shape! (We can then move these shapes back to the original locations where we really want them later.)

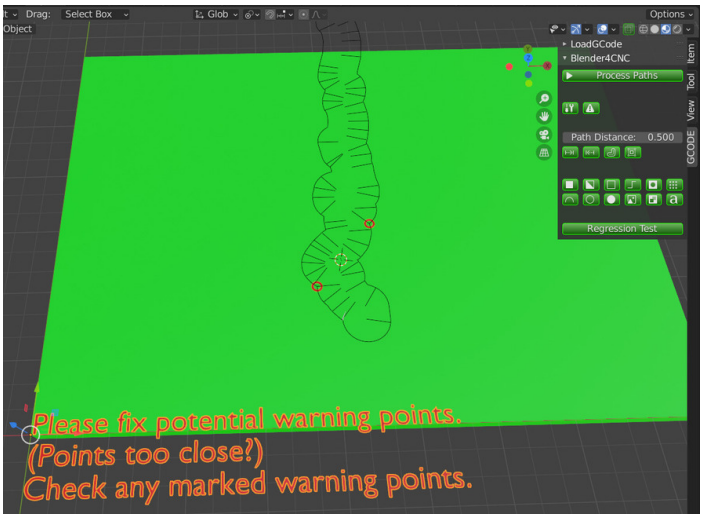

Figure 406: Try to expand this latest shape and we run into some familiar warnings that are now occurring at different points on the perimeter. If the exact precision of this shape were critically important, we could continue to fix the warnings areas each time we expand the shape. But this shape is not required to be so precise so let's start simplifying ALL of it before we perform the next expansion.

Figure 407: Switch to wireframe view and select almost all the points (just don't select the points that make up the top and bottom curves).

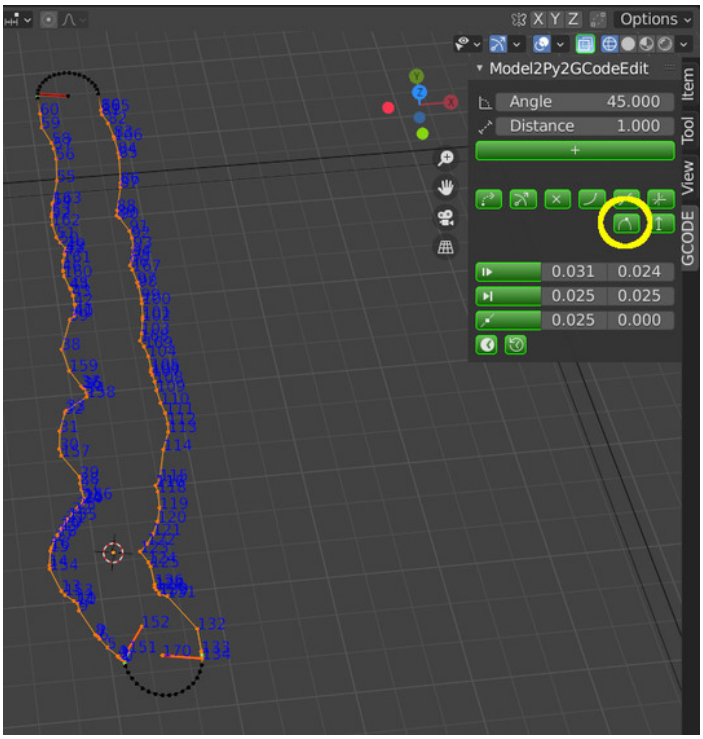

Figure 408: Click the button to Reduce curves. We don't need the complexity of all those curves. If the cutter follows this path by moving in straight lines between these new points, the end result is still going to look really good.

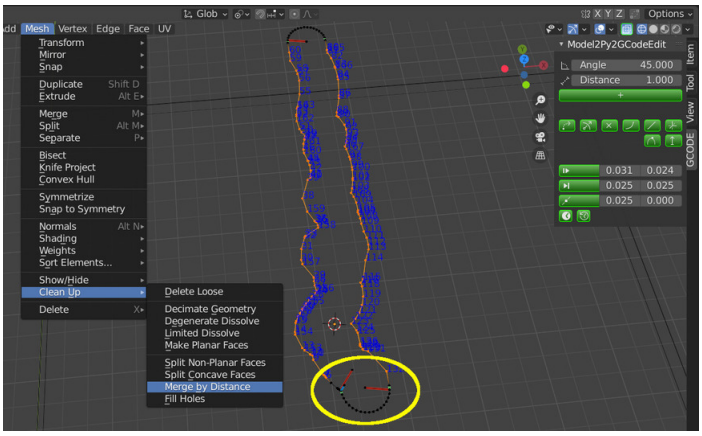

Figure 409: We really don't need the complexity of this many points either. For this next step make sure the points around the start flag are not selected. Then merge any selected points that are within 0.2" of each other.

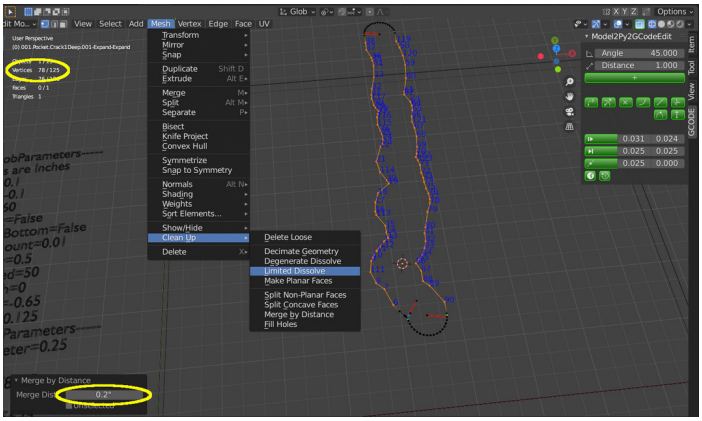

Figure 410: The number of vertrices is reduced significantly, then perform a "Limited Dissolve" at 5 degrees to further reduce the number of vertices.

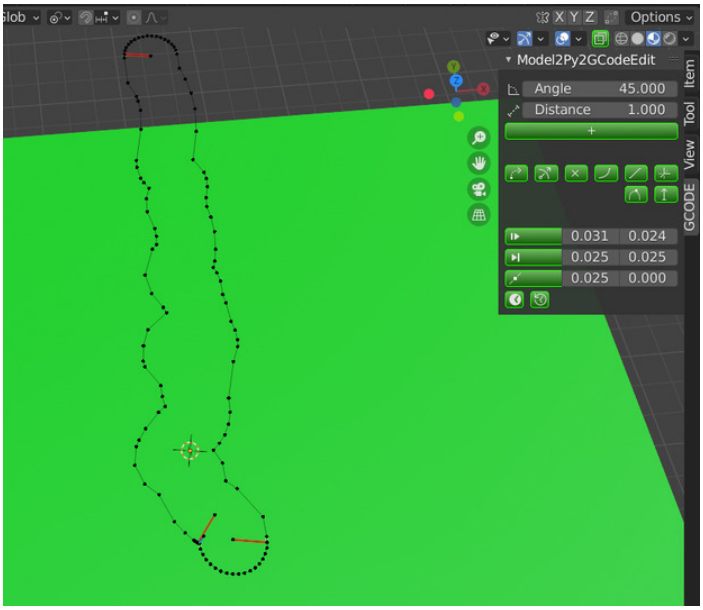

Figure 411: Performing these types of simplifications on the mesh will greatly reduce the chance of problems occurring with complex or weird shapes.

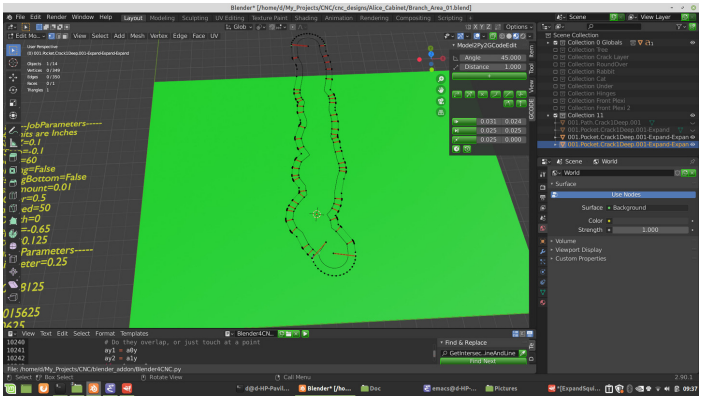

Figure 412: The next iteration of expansion. More curve segments have been introduced around each outside corner.

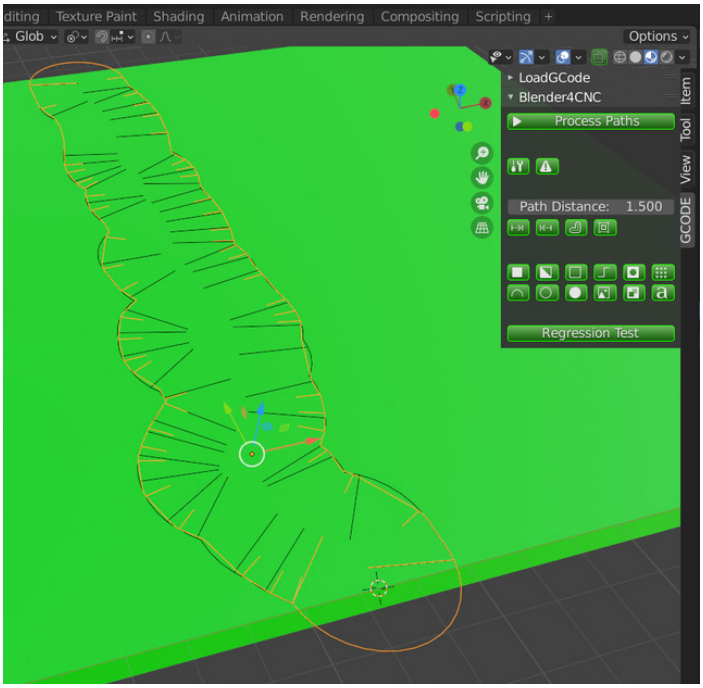

Figure 413: IF the end goal was to expand the original squiggle by 1.5", we could have done it in a single expansion. Here we show the (somewhat minor) differences between the three-step 0.5" expansions that highlighted how internal math errors accumulate in a mesh (orange) versus expansion by 1.5" (black).

1.13 Text

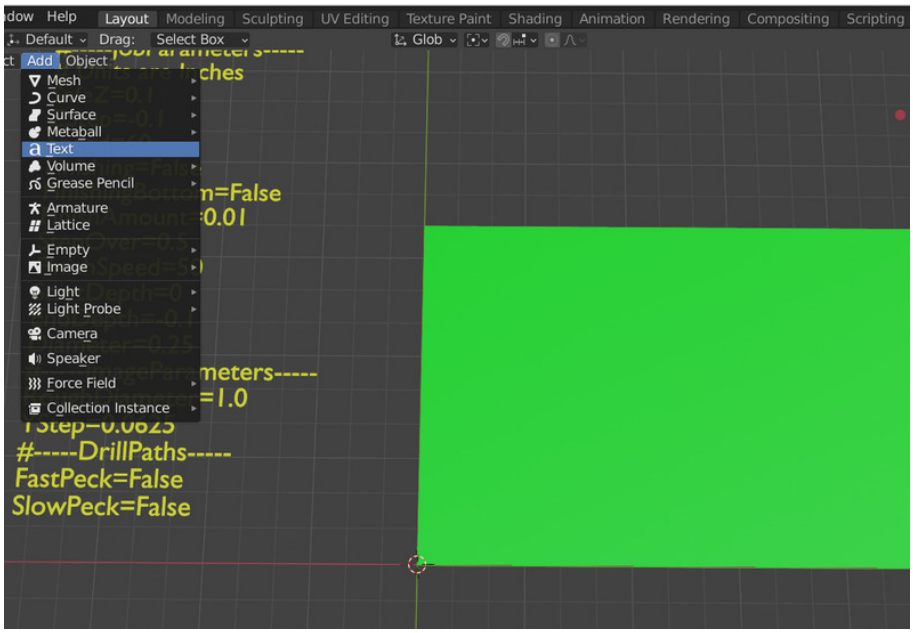

Figure 414: Blender4CNC is not great for producing text (yet) but let's go through an example of how to achieve text. Beginning with a template, go to the "Add" menu and select "Text".

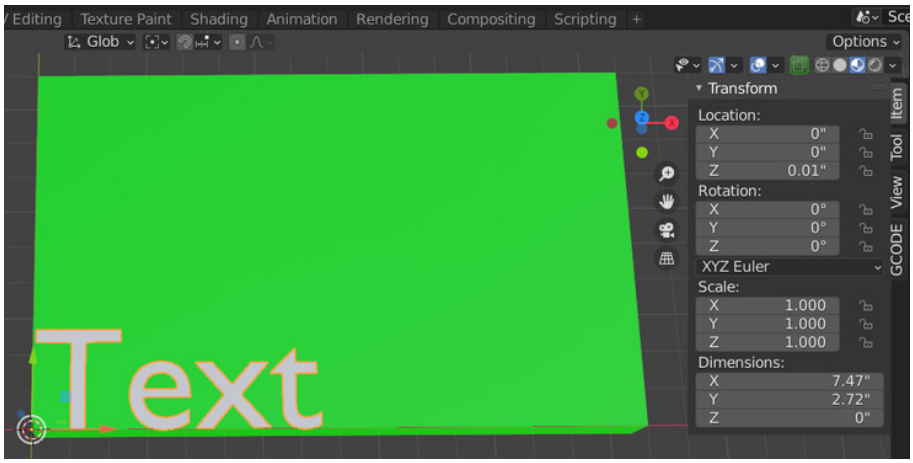

Figure 415: A text object is added containing the text "Text". I have moved the Z location to be slightly above the material.

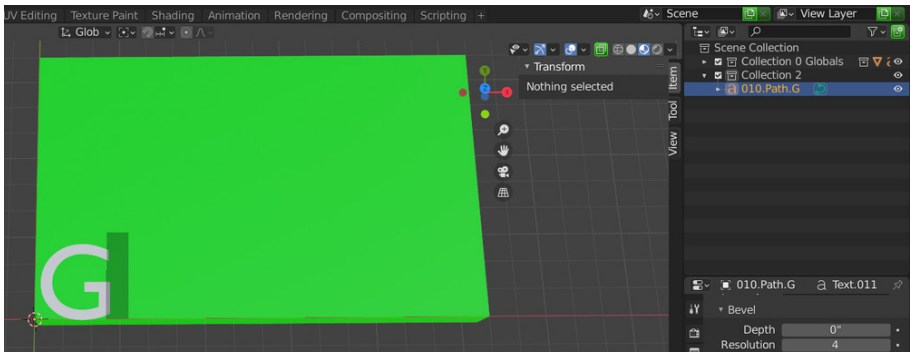

Figure 416: Rename the object to "010.Path.G" and then select the object in the viewport and press TAB to enter edit mode and change the text to "G". Then press TAB to exit edit mode.

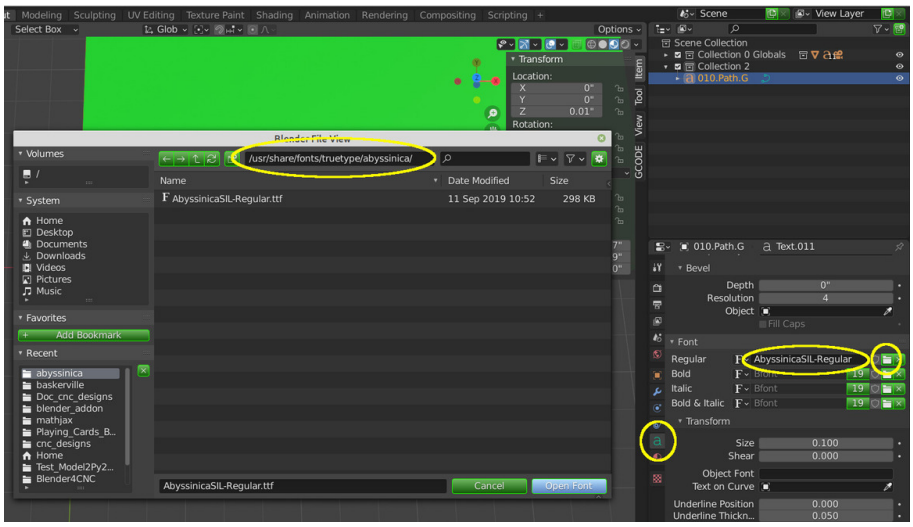

Figure 417: In Blender, you can change the font of a text object to any font that is installed in the operating system. Click on the "font" tab and then click the "Open Font" button to bring up a dialog window where you can select any font that is installed on your operating system. In this example, fonts on Linux Mint are in "*usr/share/fonts*" and I have chosen an AbyssinicaSIL font.

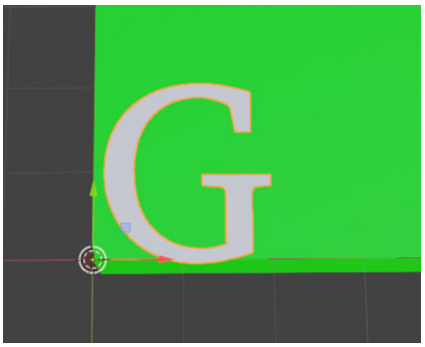

Figure 418: This is how a "G" looks in the AbyssinicaSIL font.

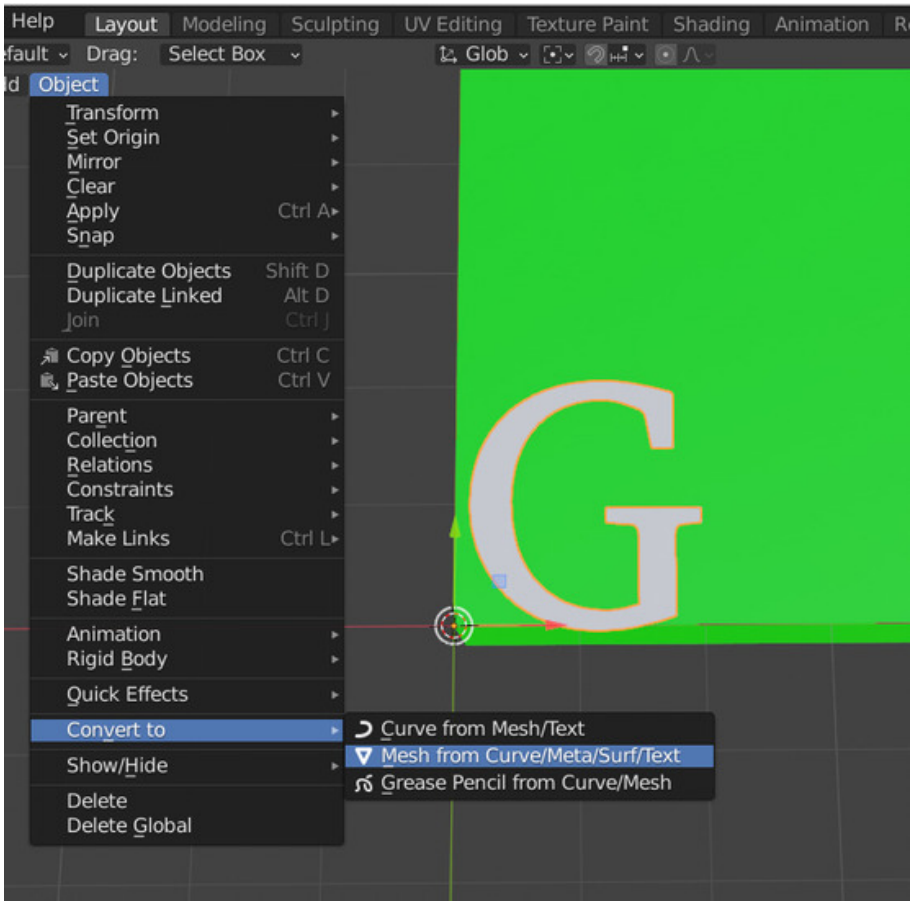

Figure 419: It is time now to convert this text object into a mesh. Go to "Object", "Convert to" and then select "Mesh from Curve/Meta/Surf/Text". It won't look immediately different.

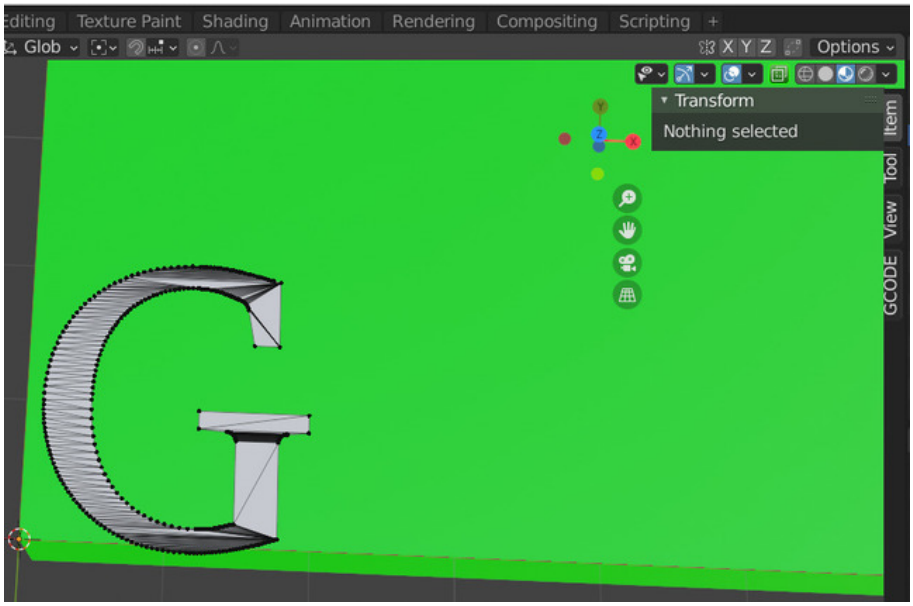

Figure 420: Press TAB to enter edit mode and this time you will see that you can no longer edit the text like you would in a word processor, instead you have many points outlining the shape of the letter "G". Unfortunately, you also get many internal edges as the "face" of the G is made up of many triangles - this is not good because CNC operations like paths and pockets **cannot** have internal edges like this.

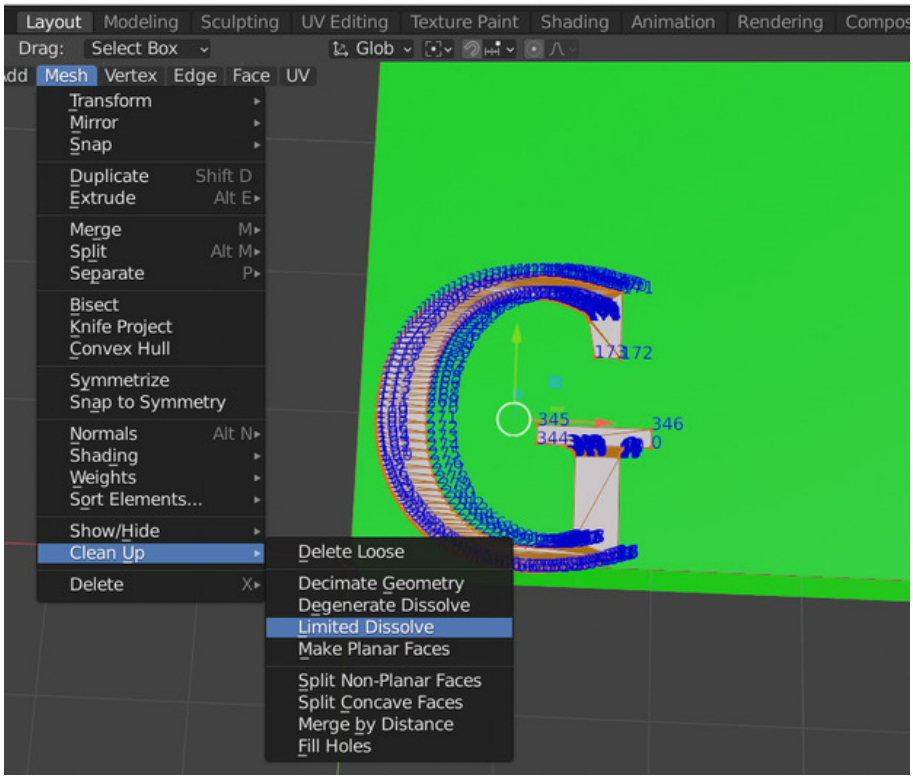

Figure 421: You can select all the internal edges and delete them but there is a quicker way by using the "Limited Dissolve" function. Select all the points and go to "Mesh", "Clean up" and then select "Limited Dissolve".

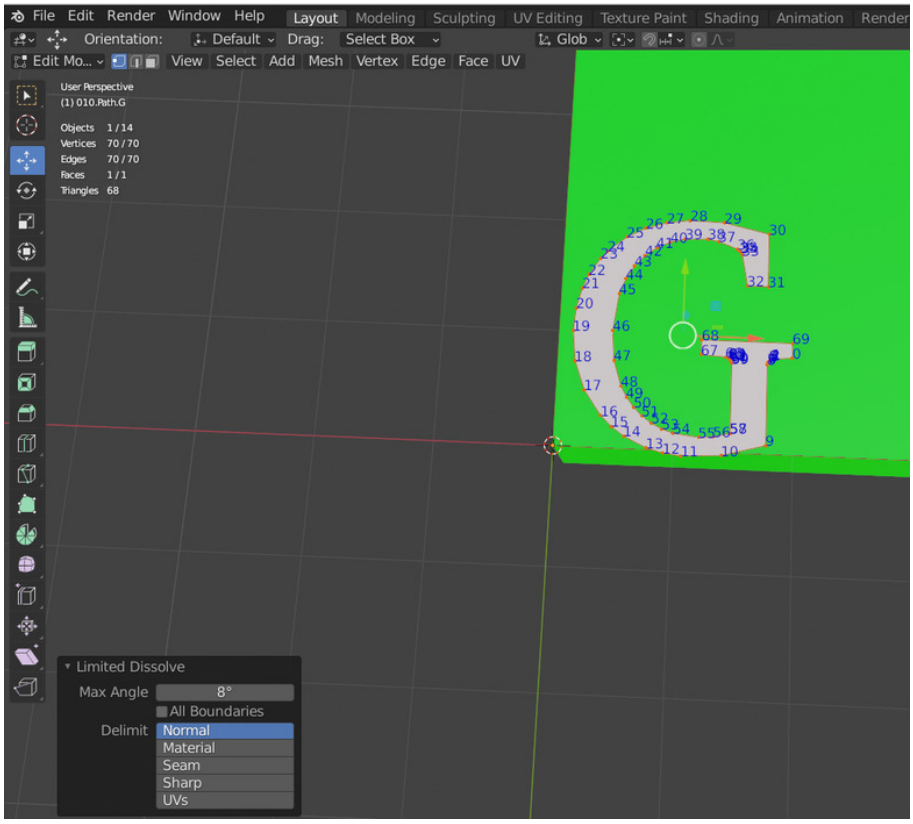

Figure 422: The higher you set the angle, the fewer points will be in the result. The really good side effect of using the Limited Dissolve function is that even if you do not reduce the number of points by much, it is that all the triangles are merged into one single face and there are **no** internal edges.

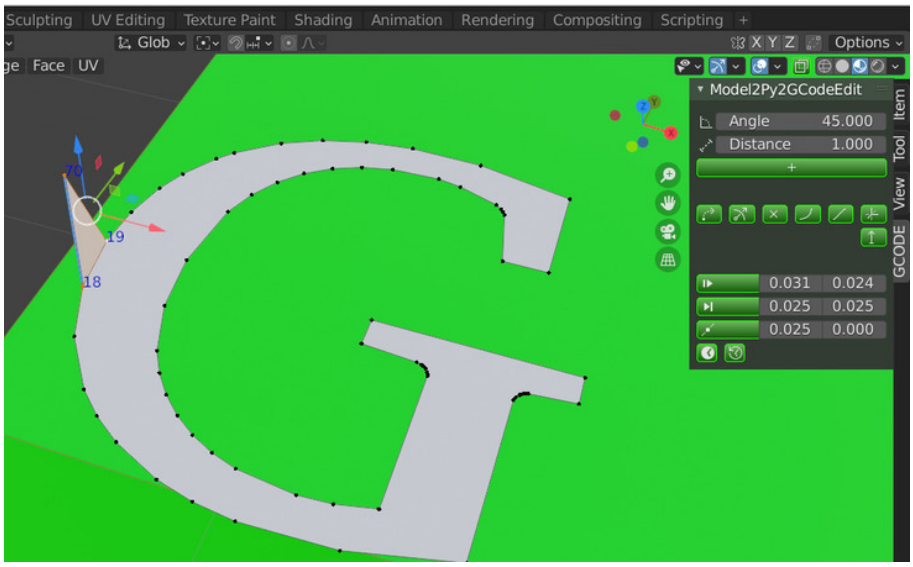

Figure 423: Now that we have a nice path around the exterior of the "G" we just need to create a "start" flag. Pick a point to be the start and create a start flag.

Figure 424: Here is the "G" path visualized with a 0.125" diameter bit.

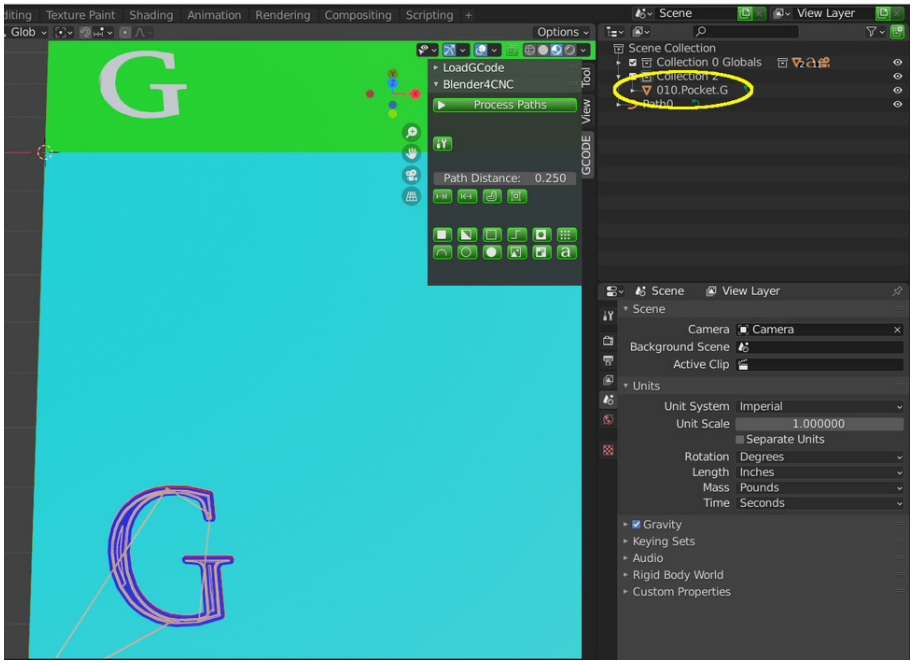

Figure 425: By changing the name to be a pocket instead of a path, we can see how the text looks as a pocket.

1.14 Loading and Visualizing Any GCode Files

The Blender4CNC addon can also be used to visualize any GCode file. However there are some GCodes that it ignores or does not (yet) understand. It should perform fairly well with GCode that is intended for a simple 3 axis CNC machine for tasks in the X-Y plane. Some of the GCodes that it ignores will have no impact on how a GCode file is visualized anyway.

Figure 426: When you click the arrow beside the "LoadGCode" text you see a field to enter a file name and the "Load GCode" button.

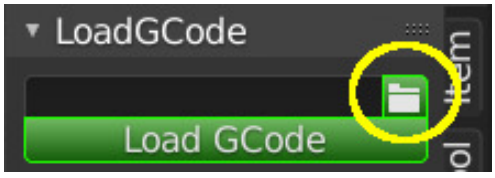

Figure 427: Make a new Blender project with "File" - "New" and then click the file browse button and browse to a valid GCode file.

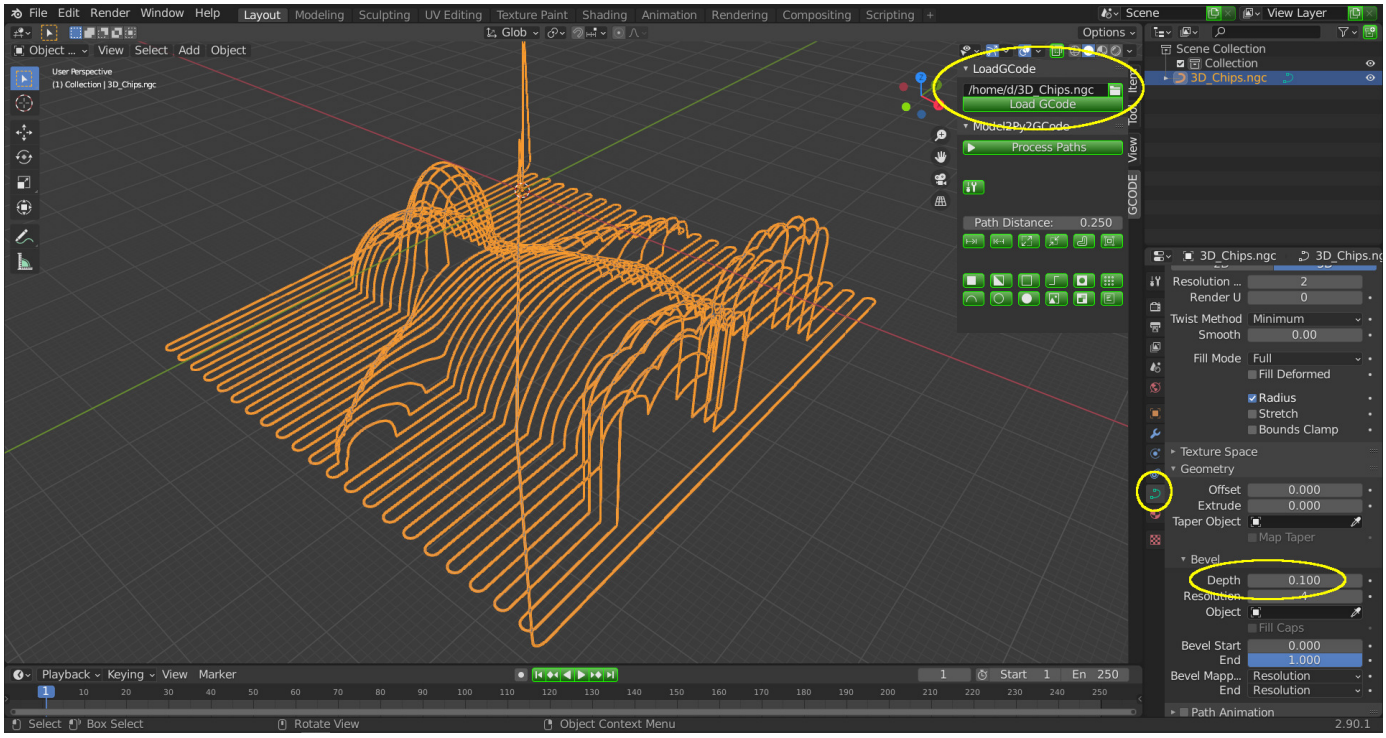

Figure 428: Click "Load GCode" to actually visualize the GCode file selected. In this example we have loaded the "3D_Chips.ngc" file supplied with LinuxCNC. To make the path more visible, select the path and then under the "Curve" tab, increase the Bevel Depth to something like 0.1. Depending on the scale of the GCode, you may need to zoom in or out to see it properly.

The "Load GCode" function of Blender4CNC was modelled on the documentation given in "The NIST RS274/NGC Interpreter - Version 3", NISTIR 6556, August 17, 2000. It also attempts to handle the GCode extensions defined/recognized by LinuxCNC (such as WHILE, ENDWHILE, IF, THEN etc.) It has been tested on numerous example GCode files that are supplied with LinuxCNC.

"Load GCode" will ignore the following codes:

- G93 and G94 (set feed rate modes).
- F (feed rate).
- S (spindle speed).
- T (tool)
- M6 (change tool)
- M3, M4 and M5 (spindle on/off)
- M7, M8 and M9 (coolant on/off)
- M48 and M49 (enable.disable overrides)
- G4 (dwell)
- G17, G18 and G19 (set active plane)
- G20, G21 (set length units)

- G40, G41 and G42 (cutter radius compensation)
- G43 and G49 (cutter length compensation)
- G54..G59, G59.1, G59.2, G59.3 (coordinate system selection(
- G61, G61.1 and G64 (path control mode)
- G90 and G91 (distance mode)
- G98 and G99 (retract mode)
- G28 and G30 (home)
- G10 (change coordinate system data)
- G92, G92.1, G92.2, G94 (axis offsets)
- G80, G82, G84-G89 and G53 (motion)
- M0, M1, M2, M30 and M60 (stop)

It successfully visualizes the following GCode files that are supplied with LinuxCNC:

- $3D_{Chips}$, b-index, cds, $comp311_2$, comp311, comp-g1, gmoccapy_2_tools_with_cutter_radius_compens plasmatest, skeleton, gridprobe, flowsnake, daisy, offsets

Currently, it does **not** successfully visualize the following GCodes that are supplied with LinuxCNC (in most cases because it is not 4th axis aware yet):

- 3dtest, comp, lathecomp, lathe_pawn, tort (does not handle planes other than X-Y)
- butterfly, systems (NURBS?)
- lathe-g76 (need tool table)
- g76 (G76 code not implemented)
- lathe_g70_71_demo, lathe_g7x_face_boring, lathe_g7x_quadrants (G70, G71 an G72 not implemented)
- threading (G33 codes not implemented)
- hole-circle, polar, spiral ('^' character not handled in extensions)
- m250, m61demo, m6demo, m6remap, touchoff (these are subroutine files?)
- mmount (helical arcs not implemented)

1.15 Blender4CNC List of Error Messages

1.15.1 Errors that occur when Loading/Displaying GCode files.

See the GCode specification - RS274NGC.

1. UNKNOWN oCMD

   The only "O" Codes understood by the GCode parser are - WHILE, ENDWHILE, IF, ELSE, ENDIF, SUB, CALL, ENDSUB and DO.

2. Changing units Imperial/Metric detected.

   The GCode contains a G20 code **and** a G21 code. The GCode parser does not allow both imperial and metric modes to be used in a single GCode file.

3. Invalid Arc (R and I,J specified)

   An arc command (G2 or G3) has specified an "R" parameter **and** has also specified an "I" or "J" parameter. In radius format, a example arc command looks like: G2 X10 Y10 R5 or G3 X10 Y10 R5.

4. Invalid Arc (none of X,Y specified)

   An arc command (G2 or G3) has specified an "R" parameter but has not specified an "X" or "Y" parameter. In radius format, a example arc command looks like: G2 X10 Y10 R5 or G3 X10 Y10 R5.

5. Invalid Arc (none of I,J,R specified)

   An arc command (G2 or G3) has not specified any "R", "I" or "J" parameter. In radius format, a example arc command looks like: G2 X10 Y10 R5 or G3 X10 Y10 R5. In center format, a example arc command looks like: G2 X10 Y10 I5 J5 or G3 X10 Y10 I5 J5.

6. Invalid Subroutine

   The GCode parser detected an invalid subroutine.

7. Invalid Arc (end point cannot equal start point in radius format)

   An arc command (G2 or G3) was specified in radius format but the start and end points are the same.

8. Invalid Arc (radius too small) rValue=?, q/2=?

   An arc command (G2 or G3) was specified but the radius is too small.

9. Invalid Parameter

   The GCode parser encountered an invalid parameter. Examples of a valid parameter are: #15 and #[10+5].

10. Invalid Real Value

    The GCode parser encountered an invalid real value. Examples of "Real Values" are a number like 10 or -25.3; an expression like [10+10]; a parameter like #15; a function like cosine[45] (allowable functions are: $\text{absolute}_{\text{value}}$, $\text{arc}_{\text{cosine}}$, arc_{sine}, cosine, $\text{e}_{\text{raisedto}}$, fix_{down}, fix_{up}, $\text{natural}_{\text{logof}}$, round, sine, $\text{square}_{\text{root}}$, tangent); or a tan function like tan[30]/[50].

11. Invalid O Expression

    THe GCode parser encountered an invalid expression for an "O" command. Examples of "O Expressions" are: [#10 EQ 5] or [#10 LT 5].

12. Invalid Expression

    The GCode parser encountered an invalid expression. Examples of an expression are: [10+10] or [#10-#9].

13. Invalid binary operator

    The GCode parser encountered an invalid binary operator. Allowable binary operators in an expression are: **, /, mod, *, and, or, xor, + and -.

14. Invalid atan function

    The GCode parser encountered an invalid atan function. An example of a tan function is tan[30]/[50].

15. Invalid function

    The GCode parser encountered an invalid function. Valid functions are: ABS, ACOS, ASIN, COS, EXP, FIX, FUP, LN, ROUND, SIN, SQRT, and TAN. An example of a function is sin[30].

16. More than 4 M commands on line

    The GCode specification allows no more than 4 M codes on a line.

17. Invalid line number

    A line number can only consist of digits (for example N10).

18. Invalid line number (exceeds 5 digits)

    A line number can have a maximum line number of only 5 digits (for example N123456 is invalid but N12345 is valid).

19. Invalid comment within comment

    Comments cannot be embedded within a larger comments. For example, "(This is (a comment))" is not allowed.

20. Code ? appears multiple times on line

    A code like "X" can only appear once on a line.

21. Incompatible Codes (G or M) on line

    Certain G codes are incompatable with each other and cannot appear on the same line. For example, G0 and G1 cannot be on the same line. Also, certain M codes are incompatable with each other and cannot appear on the same line.

22. Invalid Number

    A number can only contain digits 0-9; a leading + or - sign; and may contain a decimal point '.'.

23. Invalid character

    Valid words in GCode can only start with a letter/symbol in the following string "ABCDFGHIJKLMNOPQRSTXYZ#". (The whole alphabet **except** E,U,V and including the "3" symbol.)

24. GCode is imperial, this project is not in imperial units!

    The Blender units in the current project **must** match the units being used in the GCode file. You can only load a metric GCode file into a metric project or a project with units set to "None". You can only load an imperial GCoge file into an imperial project or a project with units set to "None".

25. GCode is metric, this project is not in metric units!

    The Blender units in the current project **must** match the units being used in the GCode file. You can only load a metric GCode file into a metric project or a project with units set to "None". You can only load an imperial GCoge file into an imperial project or a project with units set to "None".

26. UNKNOWN FUNCTION IN VisualizeGCodePath

    An unknown function was encountered. This error should never occur and posibly indicates a bug in the GCode parser code.

1.15.2 Errors that occur while creating operations

1. Cannot create path from a single point

   You cannot create a "path to the right" or a "path to the left" of the current path if the current path only contains a single point. The current path must have at least 2 points.

2. Cannot create path from a circle

   You cannot create a "path to the right" or a "path to the left" of the current path if the current path is a circle.

3. Nothing selected!

   You must select at least one mesh before attempting to use the function.

4. Too many objects selected!

   You have more than one object selected. The function can only operate on one object.

5. Please select a path only!

   The function can only be applied to a "Path" operation.

6. Please select a path or pocket only!

   The function can only be applied to a "Path" or "Pocket" operation.

7. Shape shrunk to nothing!

   The selected object disappeared entirely when it was shrunk by the given amount.

8. Please select a pocket only!

   The function can only be applied to a "Pocket" operation.

9. Selected operation must be within a collection.

   The CNC operation must be in a collection before the function will be applied.

10. Please select exactly 2 points.

    The function requires that you select exactly 2 points.

11. Please select a single point.

    The function requires that you select exactly one point.

12. Start and End points are not equal distance from Center point.

    When using the functions to create a curve, you must make sure that the start and end points are the same distance from the center point (all points on a circle are the same distance from the center and that distance is called the radius).

13. Multiple vertices at start point in mesh.

    The designated start point of the curve to be created contains more than one vertex in the mesh. (There are multiple vertices at that location and the function does not know which one to attach to for the "radius".)

14. Multiple vertices at end point in mesh.

    The designated end point of the curve to be created contains more than one vertex in the mesh. (There are multiple vertices at that location and the function does not know which one to attach to for the "radius".)

15. Multiple vertices at center point in mesh.

    The designated center of the curve to be created contains more than one vertex in the mesh. (There are multiple vertices at that location and the function does not know which one to attach to for the "radius".)

1.15.3 Errors that occur while Processing Paths

1. ERROR - poly segment ? crosses segment ? at point ?

   An operation like a pocket consists of a path of points that ends where it started and that path **cannot** cross over itself (it can touch at a point but cannot cross).

2. ERROR - poly is not clockwise!

   This error may indicate a bug in the Blender4CNC code when processing pockets. All roughing passes of pockets are to be performed in a clockwise manner.

3. ERROR - Cannot shrink poly by ?. Poly too thin?

   An operation cannot be "shrunk". Possibly it is too thin for the specified router bit diameter.

4. ERROR - Cannot shrink/expand poly by ?. Trying to shrink a poly that is too thin?

   An operation cannot be "shrunk" or "expanded". Possibly it is too thin for the specified router bit diameter.

5. ShrinkExpand2 appears to be stuck in infinite loop?

   This error occurs (possibly) when something has gone wrong internally within Blender4CNC.

6. ERROR - poly segment from ? (point ?) does NOT touch next segment from ? (point ?)

   This error occurs if the operation mesh is not continuous.

7. ERROR - poly arc segment from ? to ? (point ?) is invalid (source/destination points are not equidistant from center)

   When defining a curve as part of a path or pocket you have to make sure that the start and end points are the same distance from the center point (all points on a circle are the same distance from the center and that distance is called the radius).

8. Cannot find start point for ?

   A valid "start point" has not been defined on the operation.

9. Cannot find up edge at multi-point for ?

   When a operation contains a path of points that touch or cross at a vertex, there must be an "up" edge to indicate the direction of travel.

10. Cannot find bevel edge at multi-point for ?

    When a operation contains a path of points that touch or cross at a vertex, there must be an "up" edge to indicate the direction of travel.

11. Cannot find next direction edge at multi-point for ?

    When a operation contains a path of points that touch or cross at a vertex, there must be an "up" edge to indicate the direction of travel.

12. Cannot attach a radius line to the start of a curve for ?

    You cannot create a curve with a radius edge attached to the start of a curve. The radius line must go to one of the points inside the curve.

13. Too many edges from curve point for ?

    The mesh in the operation contains a vertex on a curve that has too many edges.

14. Cannot find center of curve for ?

    The operation appears to define a curve which has no valid center point (or radius defined).

15. Invalid curve - end points not equidistant from center for ?

    When defining a curve as part of a path or pocket you have to make sure that the start and end points are the same distance from the center point (all points on a circle are the same distance from the center and that distance is called the radius).

16. Cannot find Project Parameters?

    A Blender4CNC project expects to have a text object named "ProjectParameters".

17. Project Parameters does not contain ? parameter!

    A Blender4CNC project expects the "ProjectParameters" text object to contain specific parameters (for example, "Version", "MsgHeight", "PathBevelDepth", "MsgZOffset", and "PathZOffset").

18. Cannot find GCode PreAmble Text?

    A Blender4CNC project expects to have a text object named "GCodePreAmble".

19. Cannot find GCode PostAmble Text?

    A Blender4CNC project expects to have a text object named "GCodePostAmble".

20. Cannot find Job Parameters?

    A Blender4CNC project expects to have a text object named "JobParameters".

21. Collection 0 Globals is not visible - do you need to click the eye icon in the outliner window?

    The collection "Collection 0 Globals" must be enabled and visible.

22. Nothing to program? Are the paths/collections visible and correctly named?

    There are no valid CNC operations to process. Maybe none are defined. Maybe some are defined but are not visible, not correctly named, or produced an error.

23. Cannot find fast peck for drill cycle

    The Job Parameters do not specify the parameter "FastPeck".

24. Cannot find slow peck for drill cycle

    The Job Parameters do not specify the parameter "SlowPeck".

1.15.4 Errors that should never occur and they indicate an internal error (bug) within Blender4CNC

1. ERROR - ShrinkExpand2 cannot process poly
2. ERROR - poly segment from ? at end point does not touch next segment ? at start point
3. ERROR - RemoveStubsAndSuperfluous seems to be stuck in infinite while loop? near ?
4. CutPocketRoughFinal stuck in infinite i loop?
5. CutPocketRoughFinal stuck in infinite j loop?