G-code with Grasshopper®

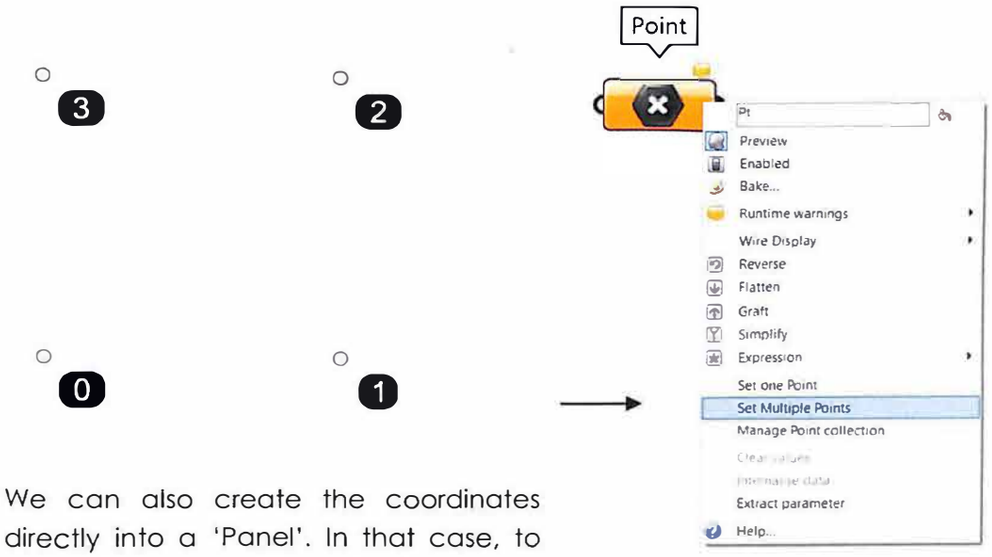
As we have already explained, the base for the g-code is made of points' coordinates. We could have the points we need for the movement of the machine but typically, we will have a polysurface, a mesh, or luckily, a polyline instead. Depending on the input object, we must follow different paths to transform it into XYZ coordinates. These are the most common paths to be followed depending on the input objects that we desire to 3D print:

			Points	→	X,Y,Z
		Polylines	→ Points	→	X,Y,Z
		Curves	→ Points	→	X,Y,Z
	Mesh	→ Polylines	→ Points	→	X,Y,Z
BRep	→ Mesh	→ Polylines	→ Points	→	X,Y,Z
BRep	→ Curves	→ Polylines	→ Points	→	X,Y,Z

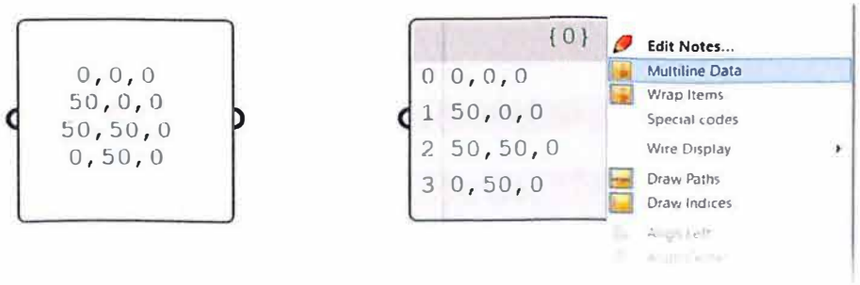
Points -> X,Y,Z

Points are the basis in understanding how to create a g-code in Grasshopper®. The most basic g-code can be created just using points in Rhinoceros® or Grasshopper®. The sequence of points' coordinates will design the movements for the 3D printer. So not only the coordinates are important but also their **order** on the list, as the movement will follow that sequence.

For the first example, the points are created in Rhinoceros® and then referenced into Grasshopper® with the 'Point' component:



We can also create the coordinates directly into a 'Panel'. In that case, to generate a proper list we should disable the option 'Multiline data':

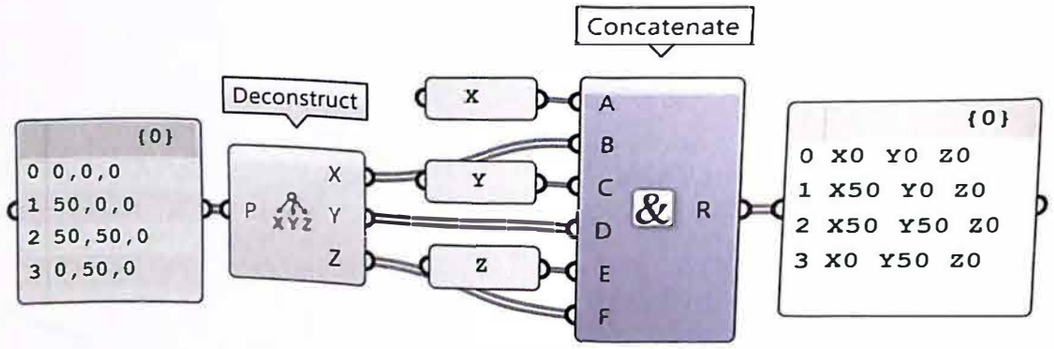


To create the g-code we need to transform the points into a text file. For that, we will use the tool 'Concatenate' on the text menu. 'Concatenate' can join letters and numbers to create text lines. It is also one of those tools with the possibility to add more inputs if we zoom in on it.

The text we have to create has this structure:

G1 F1800 X50 Y50 E5

First, 'Deconstruct' to split their coordinates. Then, add the letters and the whitespaces necessary to construct that text line:

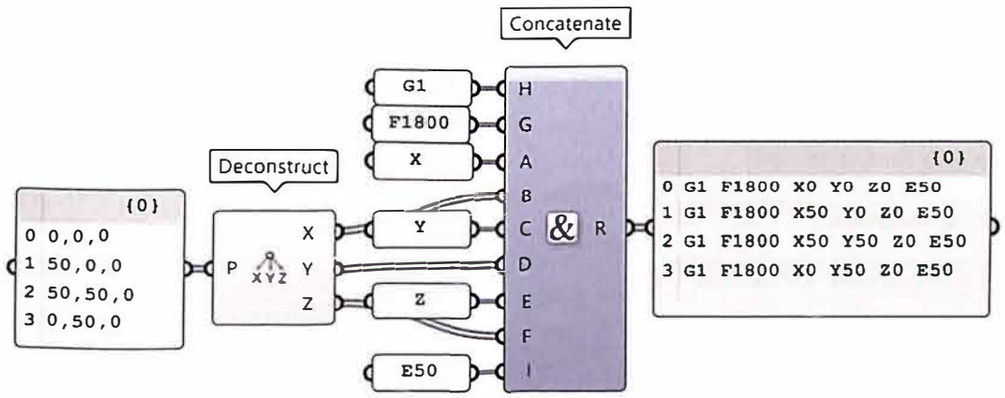


Notice that there is a whitespace (␣) in the panels before Y and Z: ␣Y ␣Z

The Z component is not usually a predominant part of the g-code as commonly 3D printing is made by layers. If the designed curves are not in a plane parallel to World XY plane and as such the points are in a curve with different heights, it is necessary to specify the Z coordinate for each point. See example at chapter *Isocurves*. Anyway, we will keep the Z coordinate for the explanation.

To get a first rough g-code we need to add more parts:

- The order G1 to indicate 'movement'
- The order F for the speed
- The letter E to tell the extruder how many millimetres of material it has to extrude:



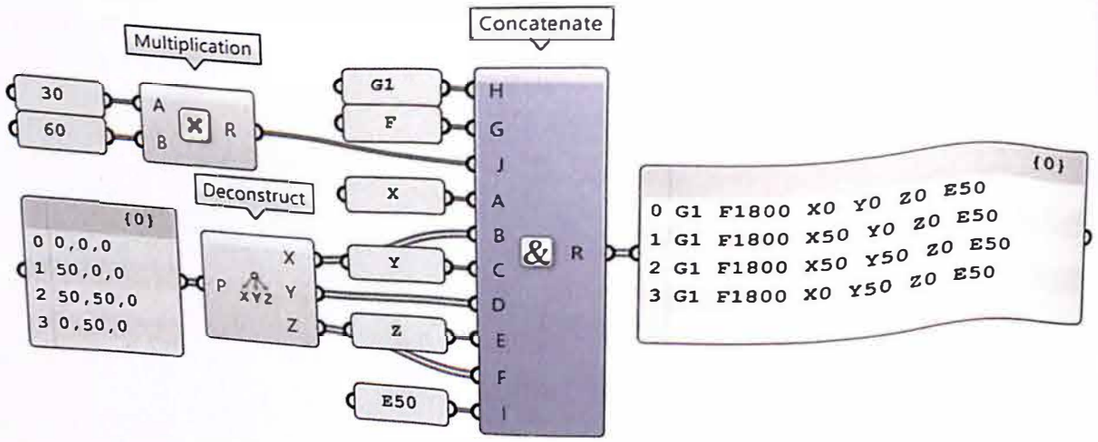
Notice the new whitespaces (␣) added to the panels. Before and after F1800 ␣F1800␣ and before E50 ␣E50.

This is the rough base to create the CORE of a g-code for a 3D printer. Now we have to add the START and the END protocols of our particular 3D printer and control the parameters F and E.

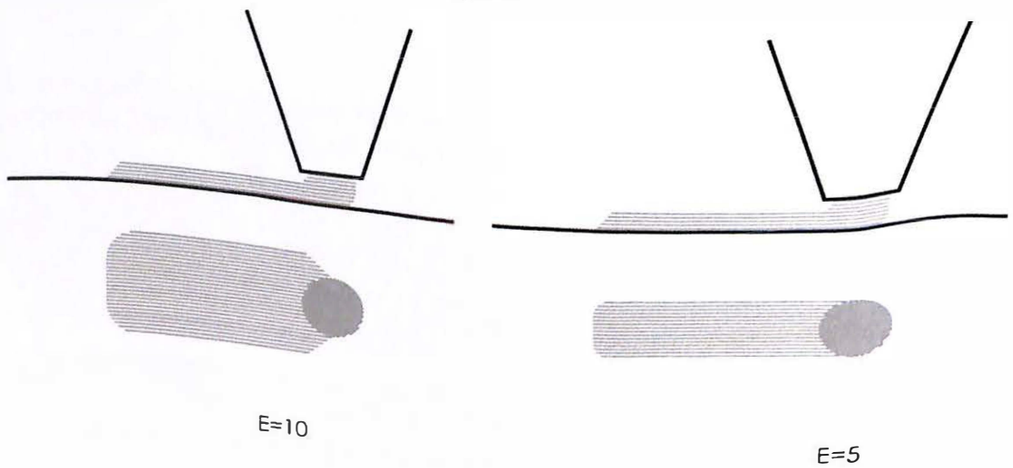
F (feed rate or speed). It is a parameter that represents the feed rate in mm/minute. F1800 mean 1800 millimetres per minute or equal to 30 millimetres per second (divide 1800 by 60 seconds in a minute). The proper speed will depend on parameters such as the layer height, material, its fluidity and even the geometry of the design. Normal speeds are between 600mm/m and 3600 mm/m but this could be higher or lower. Some tests on different speeds need to be done by the user, to understand the expected results.

A high speed for travel movement is usually used at the beginning such as for 'go home' and then, a slower one, for printing. Some materials such as translucent clay or porcelain could give interesting results varying the speed according to some pictures or attractors.

As the value for the speed is easier to understand divided by second than by minute, some users prefer to use mm/second. It is interesting to have it apart from the rest of the definition as in the figure (30*60=1800):



E (extrude rate). It is a parameter that tells the extruder how much to extrude. The parameter represents the number of millimetres extruded from the current position of the machine to the next position. The most appropriate number for the extruder will depend on the speed (F), the layer height, the material fluidity and the geometry, according to the desired result. A bigger extrusion of material will give as result a wider wall on the object.

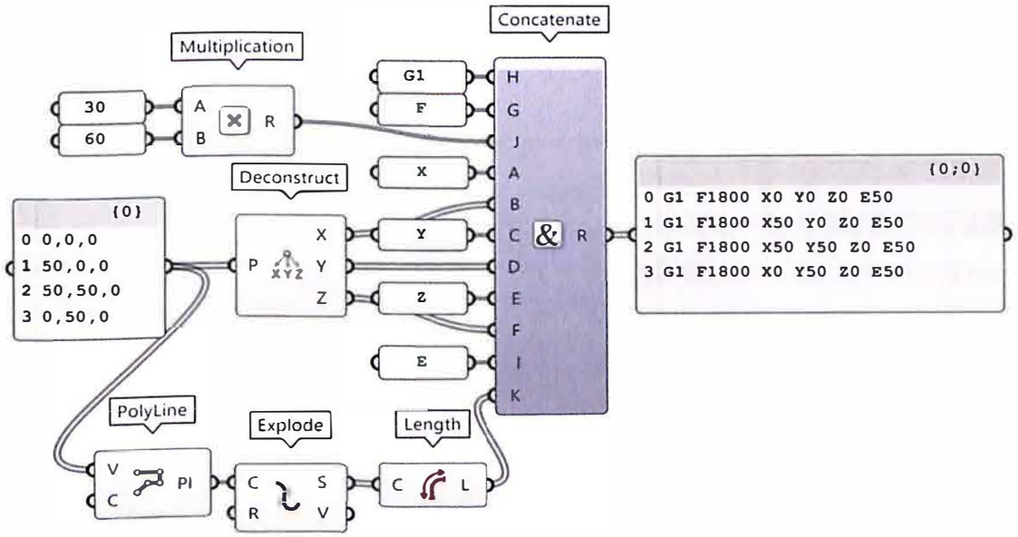


E=10

E=5

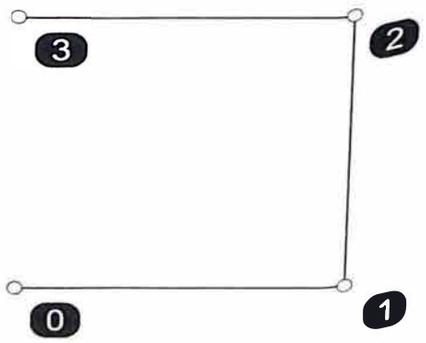
It is common to use a constant extrusion rate but depending on the geometry and a variable layer height the amount of extruded material could be different. See example at chapter Isocurves.

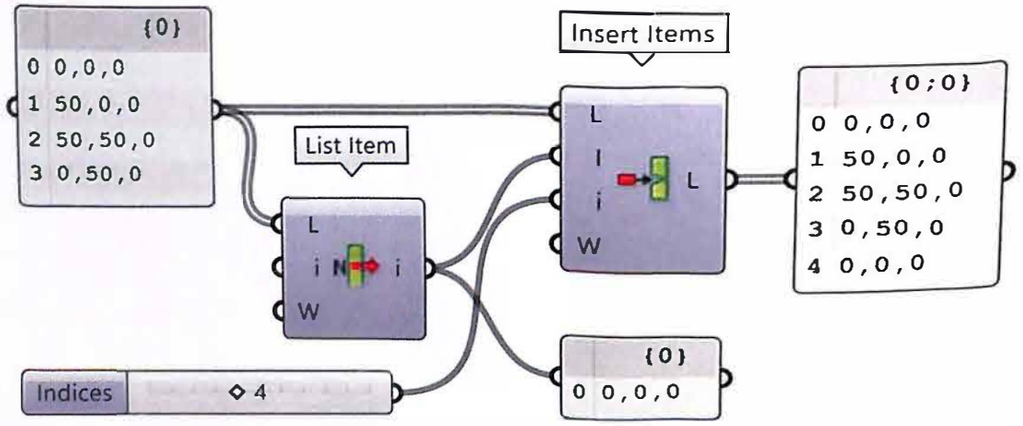
If we opt for a constant extrusion of material, the parameter E should be calculated according to the distance between points. In order to keep it simple use 'Polyline' to connect the points and 'Explode' it into their segments to get the 'Length' of each segment. If the value for the length applied to the extrusion suits our design, then we have finished. If it is too big, we shall divide it, and if it is too low, we can multiply it. Add it to the previous definition and change the previous E50 by just E.



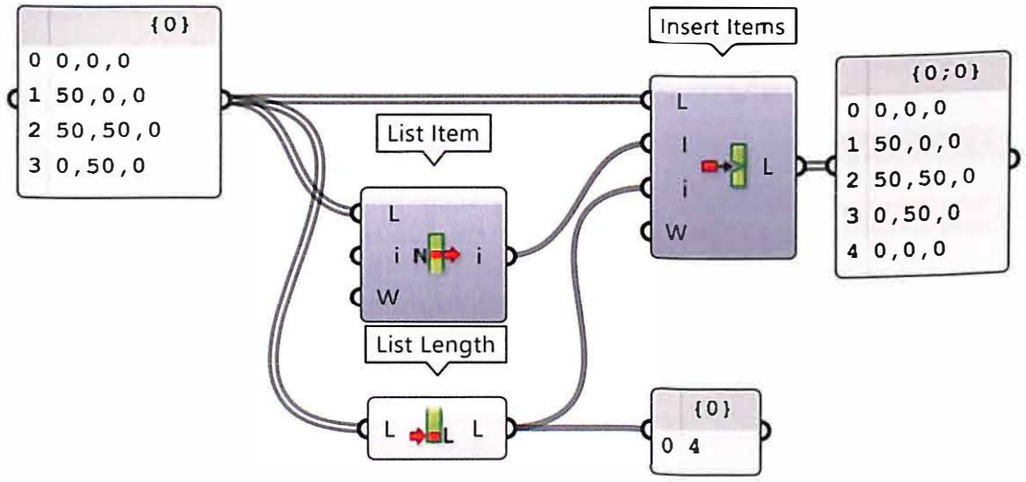
Reaching this point, we notice 4 facts:

1. 'Polyline' component displays the path for the extrusion. This is very useful.
2. The square is not closed. We can extract the first point with 'List item' component and add it at the end of the list with 'List insert' making the list one point larger. This operation will close the path.

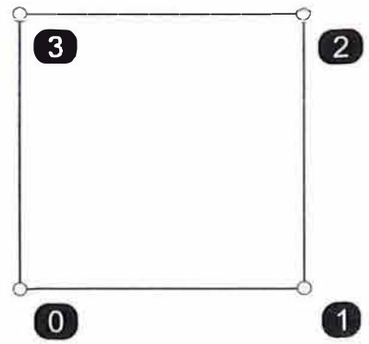




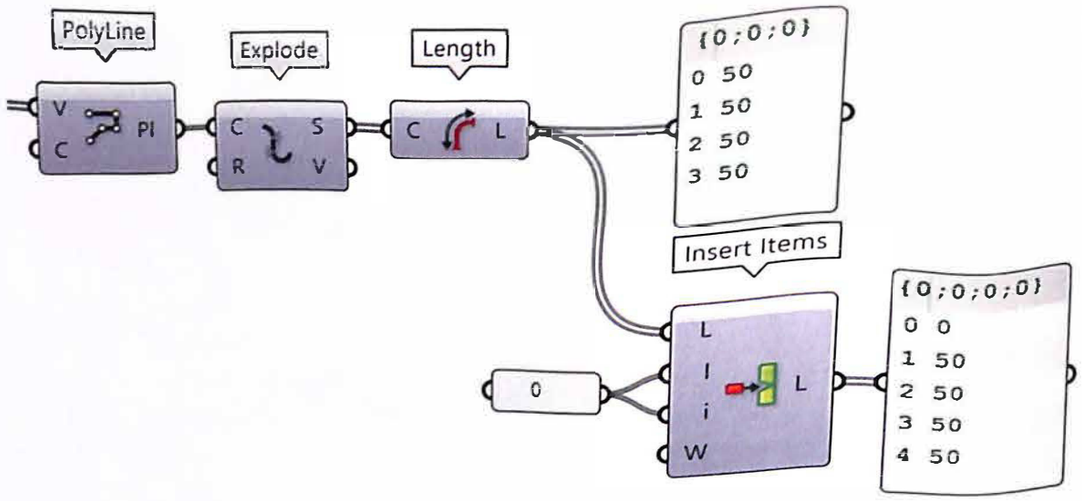
Grasshopper® users that are familiar with lists tools could also use 'List Length' to feed the input 'Indices' of 'Insert Items'. 'List Length' outputs an integer corresponding to the number of items in the list. In this case '4'. That '4' can be used for the new index.



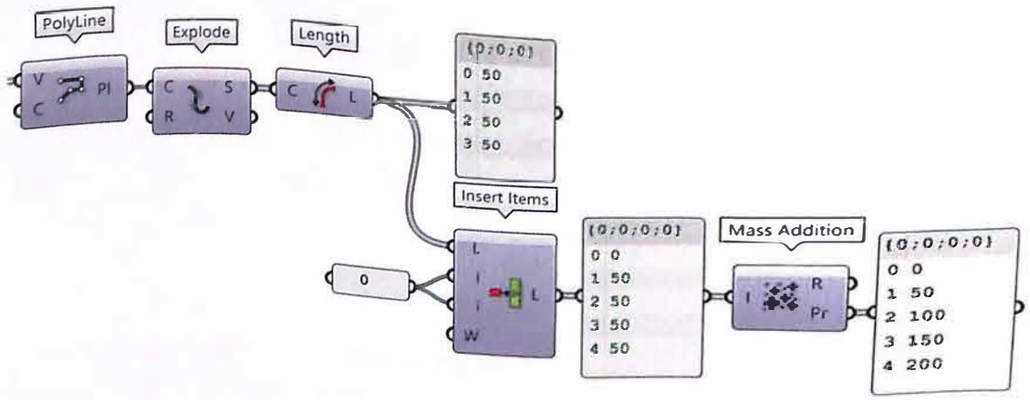
'Insert Items' adds items to a list. In the example, we are extracting or isolating '0,0,0' coordinate from the first list with 'List Item'. This component can isolate one or more elements in a new list according to their indices in said list. By default, the index is '0' so the output will be the item or coordinate with 'index 0': '0,0,0'. That output feeds the 'item' input. The 'indices' input is fed by 'List length' with a '4'. That means that '0,0,0' will occupy the index '4' in the list as displayed in the last panel.



3. The values for E. It will start extruding 50 at '0,0,0'. It should extrude 0 instead, so it should display E0 at the first line of code. Use the same strategy with 'Insert Items'. This time we will insert a '0' value at the beginning of the list of values from 'List Length':

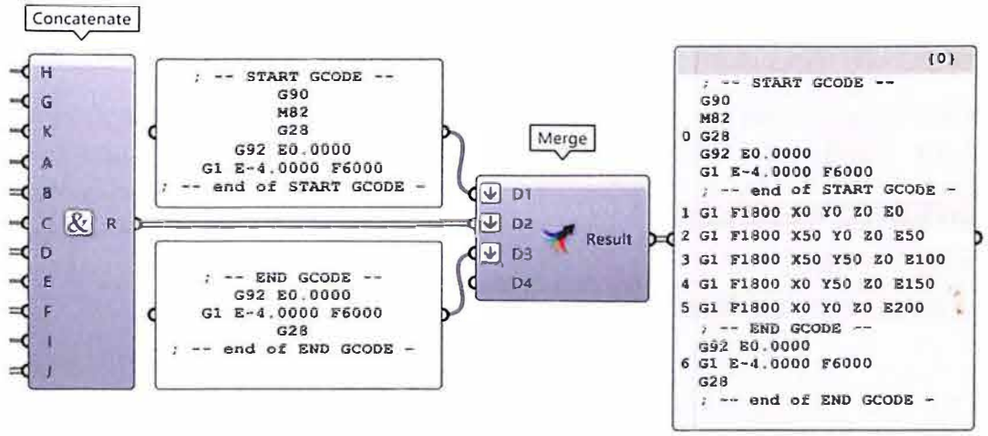


4. According to the new list of values for E, 50mm of material will be extruded from the first point (0,0,0) to the second (50,0,0) but nothing else to the rest of points as 50 remains as the only value. It is necessary that a component is able to add the lengths of the segments of the path: 'Mass Adittion'. Notice how the 50 mm. length of each segment is added to the previous length obtaining the series: 0, 50, 100, 150, 200.

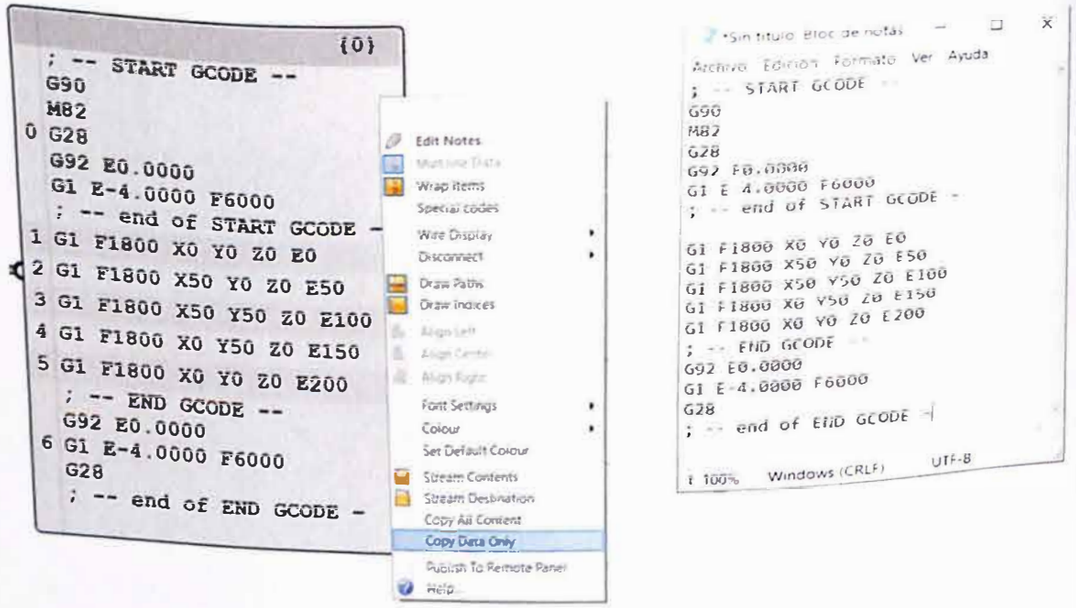


Congratulations!

You have created the Core for the g-code. Now it is necessary to add the Start code and the End code. We can do it with 'Merge Tree' tool. As 'Merge Tree' takes into account the paths of the lists to rearrange them, the inputs need to be flattened. This will make the path for each list, {0} and it will keep the order in the list as they are connected to the inputs.



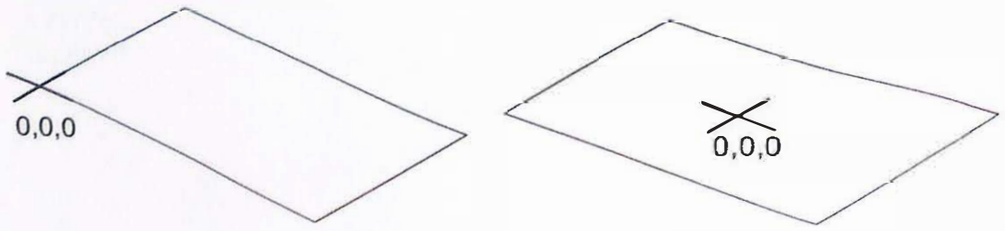
Final step, with right button over the panel, 'Copy Data Only', and paste it in 'Notepad' app in Windows or 'Textedit' in Mac. The type of file must be 'All file types'. The name of the file could be whatever, but try to avoid symbols or whitespaces. Type *.GCODE as extension.



Depending on the printer, you could save the file in an SD card, USB memory stick, or directly send it to the printer by USB connection. The path can be checked out in the slicing software as some of them have the ability to display it just using the gcode file.

...and 3D print!

There is one more important fact regarding the position of the points refered to 0,0,0 in Rhinoceros®. Deltabots 3dprinters have its own 0,0,0 in the center of the build plate, cartesian 3D printers on the bottom left corner. To be sure, you could know were it is with an existant .gcode file, reading the coordinates.



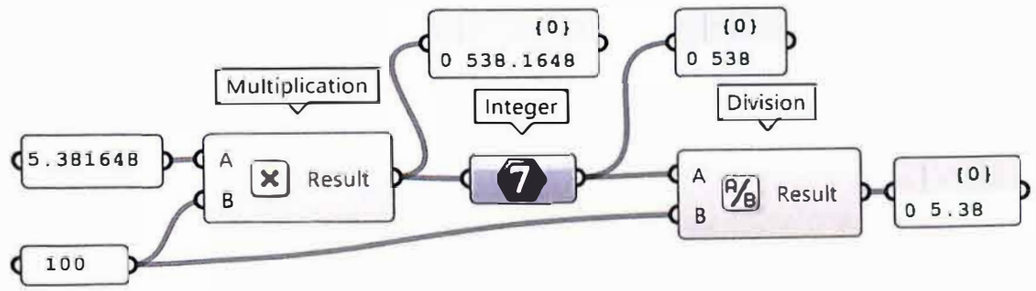
Advanced Grasshopper® users will easily make a definition to move the 3D object from its centroid to the best place for 3D printing in Rhinoceros® according to the 3D printers 0,0,0 position. Otherwise it can be moved manually in Rhinoceros®.

Important to note:

Grasshopper® uses 6 digits when there is a decimal number. E.g. 5.381648 As we work in millimetres, 6 digits is too much accuracy. 2 or 3 can be enough.

If more than 6 digits are necessary after the decimal point, Grasshopper® uses scientific notation. E.g. 5.3816e+5. This can create conflicts when reading the g-code.

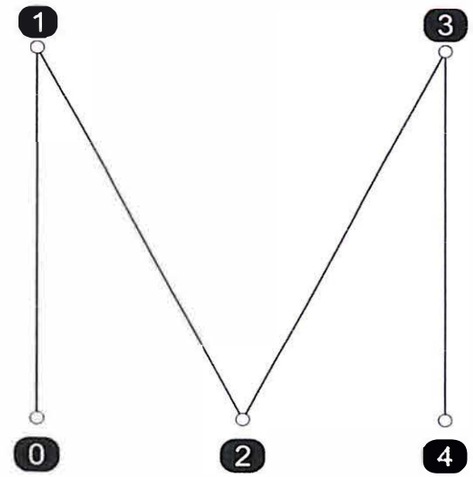
To remove digits after the decimal point, we can use a simple mathematical expression that must be applied after 'Deconstruct Point', 'Length' and any component able to create this conflict:



This definition works because 'Integer' component rounds any number to integer numbers.

If there are more than 999.999 values also on the left side of the decimal point (big projects at the E value) it can cause conflicts too. It is recommended to use then relative coordinates instead of absolute ones.

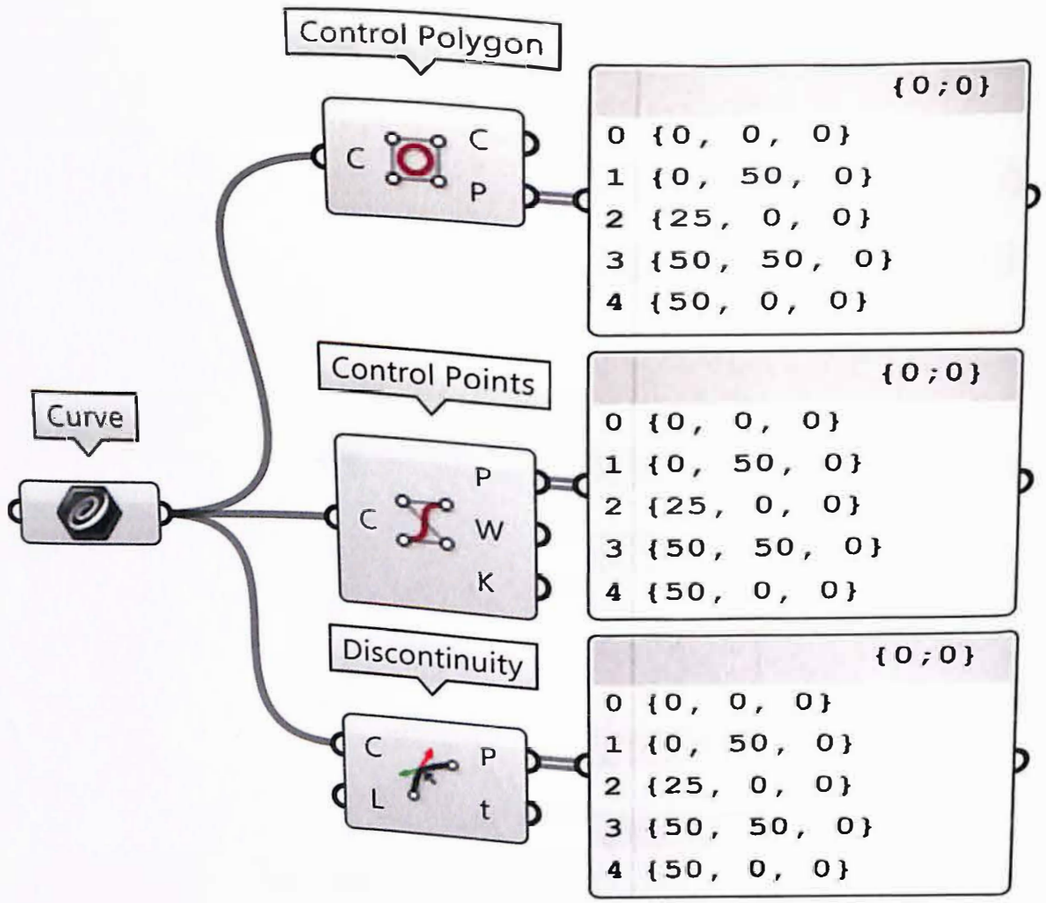
Polylines -> Points -> X,Y,Z



It is quite common to draw the polylines that define the path for 3D printing.

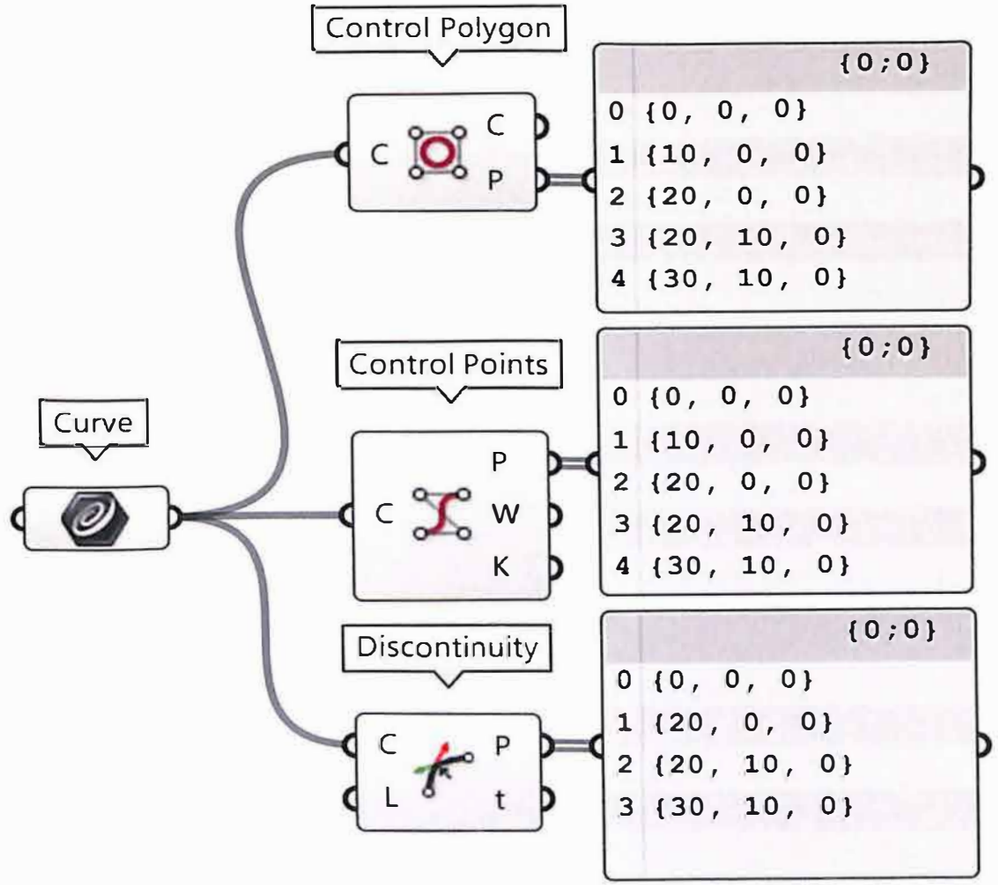
With the previous definition (from point to g-code), it is very easy to extract the points' coordinates from a polyline. There are several tools that we could use to find points in a polyline:

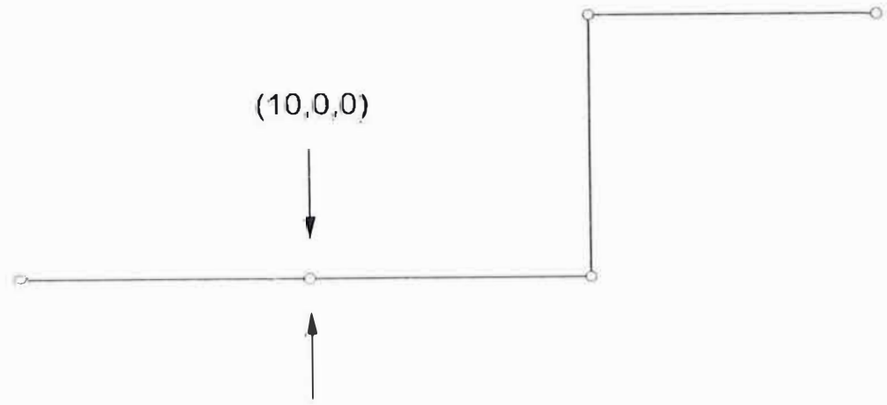
'Control polygon', 'Control points' and 'Discontinuity'



The polyline with 'M' shape is inscribed in a 50 millimetres square. The three tools seem to output the same results. But depending on the shape of the polyline, there are two main differences:

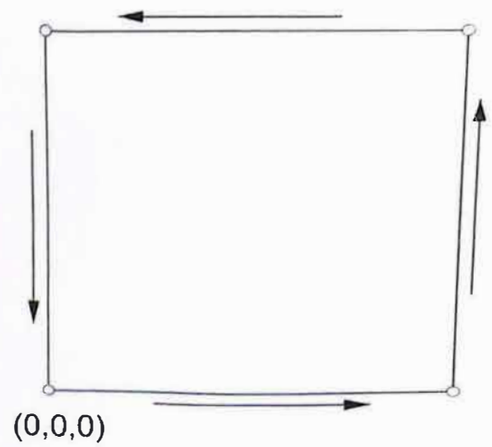
1- 'Discontinuity' will not find all the polyline's vertexes, only the discontinuities of the function. In a polyline with more than one point on a straight part, the vertexes will be omitted by 'Discontinuity', as the function is continuous, but not by the other tools as you can see in the panels on the following example:

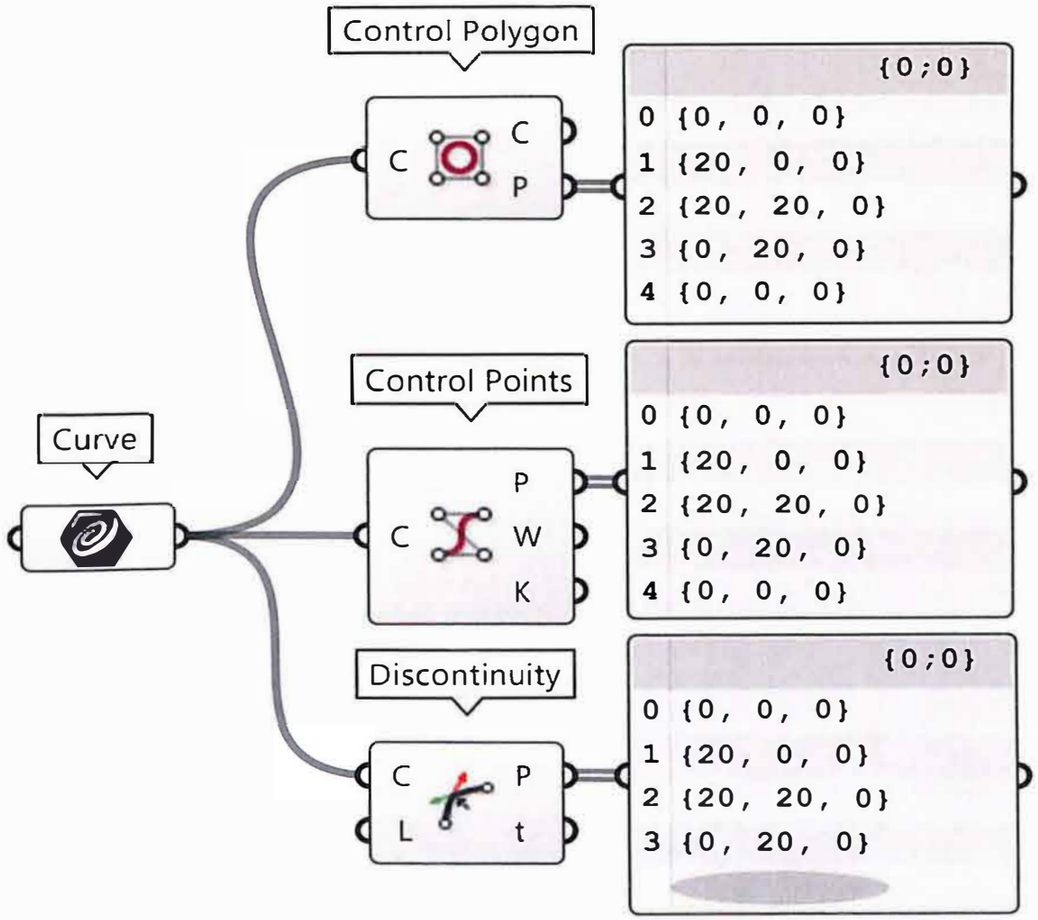




That 'Discontinuity' property will have a more visible effect on the next unit: *curve → points → X,Y,Z*

2. In closed curves, 'Discontinuity' does not repeat the seam point. The seam is both the first and the last points. It reads both as just one, the first one. It can be useful to connect to another closed curve as we will explain later. However, not so much with closed curves as we should manually repeat the first point and add it at the end of the list as explained in the previous chapter (*points → X,Y,Z*). The other two tools seem to be more useful for closed curves. Check out this square of 20 millimetres side:

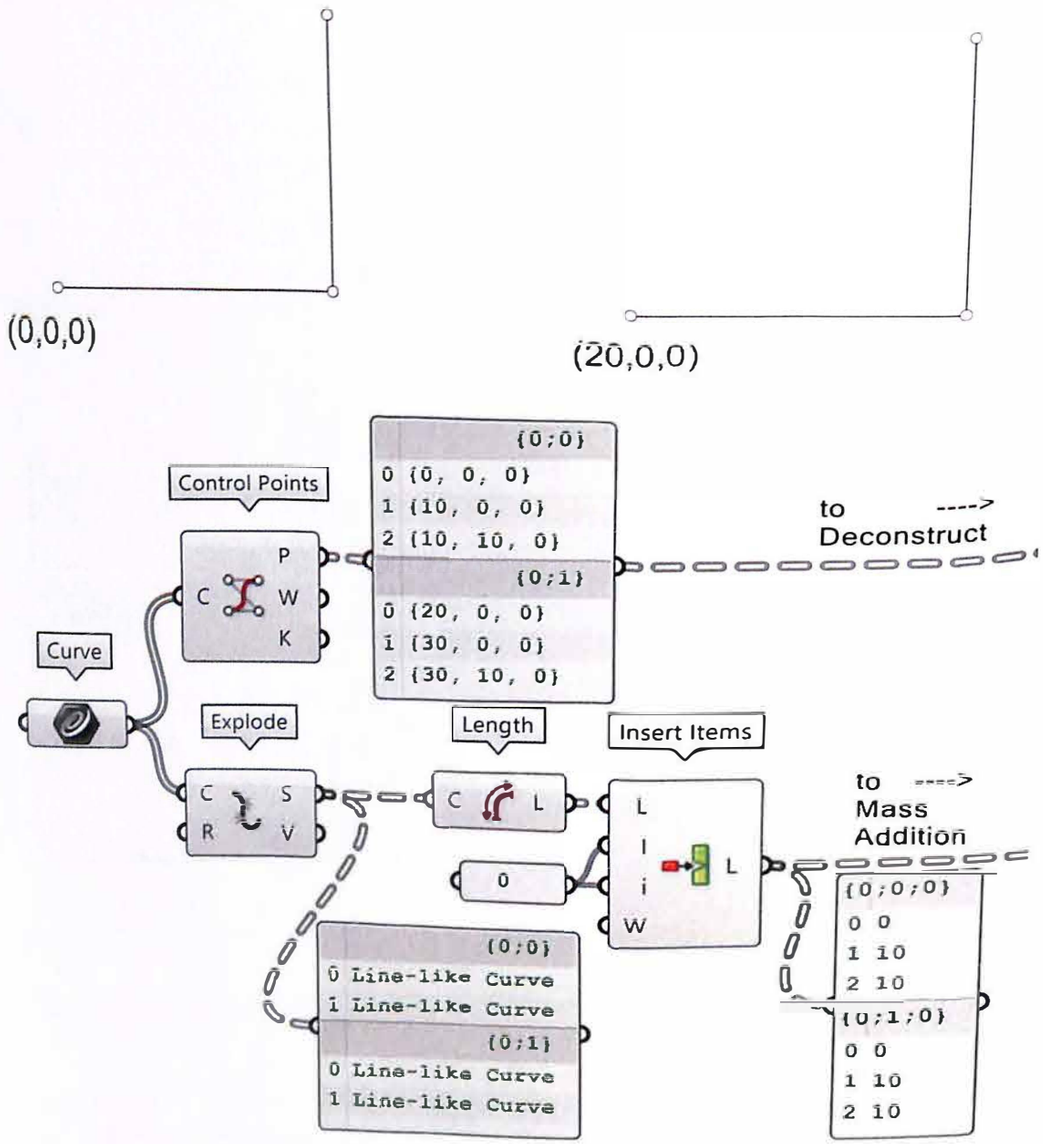




Once we obtain the desired points we can move to the g-code creation as explained in the previous chapter.

If there is more than one curve, open, closed or both, we have to detail the path to improve the results of the clay extrusion and order the code were to stop extruding (if our machine/extruder has the option to). WASP clay extruders have the option to stop and/or retract, the same as thermoplastic filament extruders. But other printers might not share that option. In that case we should look for a continuous path. It means, we should transform several curves into one.

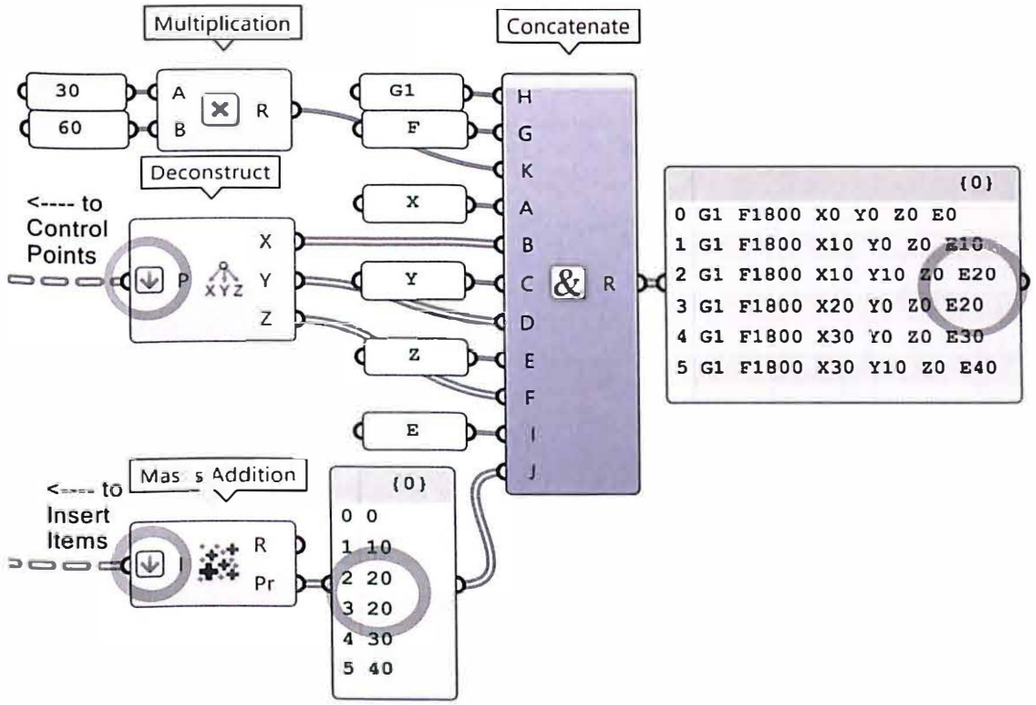
Example with two open curves in XY:



In this case there are two polylines. 'Control points' create a list with each of the polyline vertexes. 'Explode' outputs the segments for the E values. Data is organized into lists by polylines. This is very helpful for the insertion of the 0 value

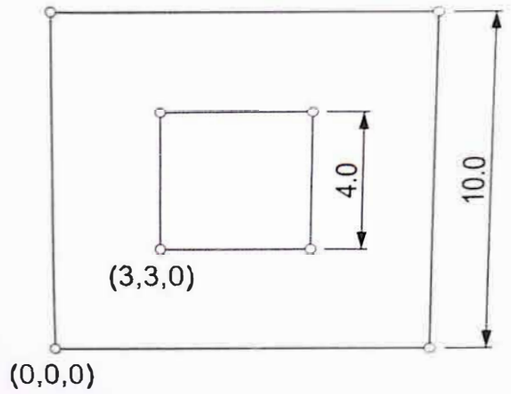
at the beginning of each polyline. Then, a 'Flatten tree' must be done to have just one single list to create the 'Mass addition' for the final E values.

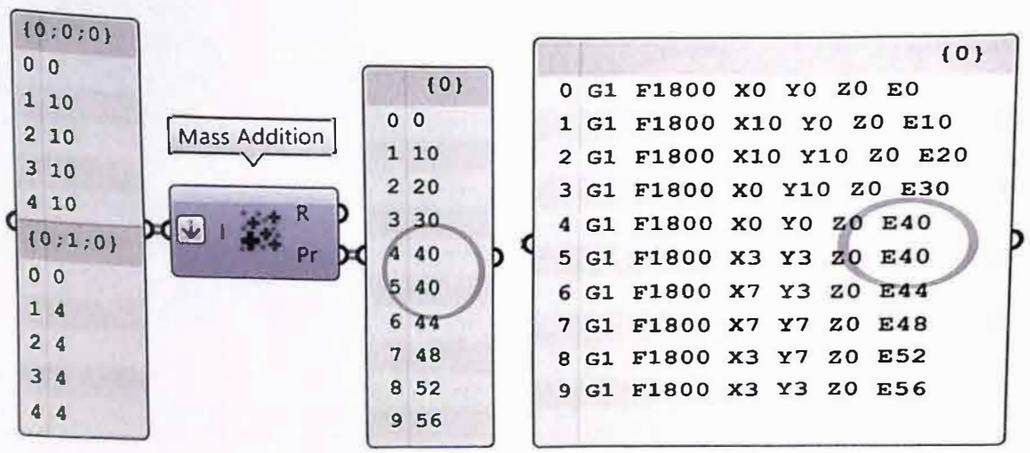
Notice that number '20' is repeated thanks to the insertion of 0 at the beginning of each list. This means that the extruder will not extrude material from the end point of the first polyline to the start point of the second:



G1 F1800 X0 Y0 Z0 E0
G1 F1800 X10 Y0 Z0 E10
G1 F1800 X10 Y10 Z0 E20 ←
G1 F1800 X20 Y0 Z0 E20 ←
G1 F1800 X30 Y0 Z0 E30
G1 F1800 X30 Y10 Z0 E40

This definition also works with more than one curve or even with closed curves:

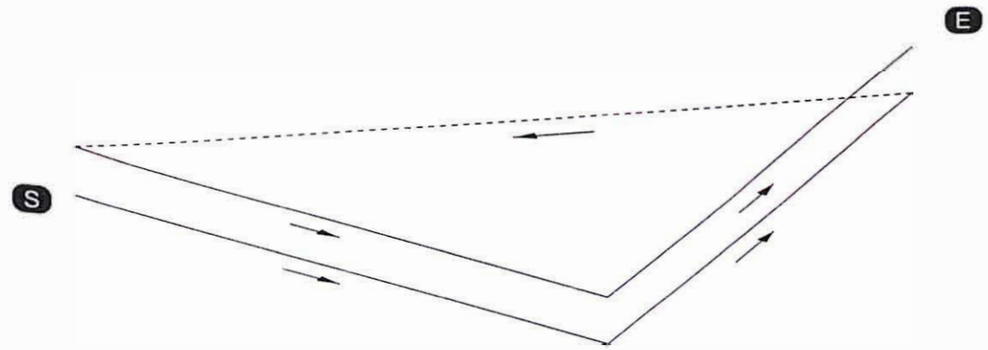


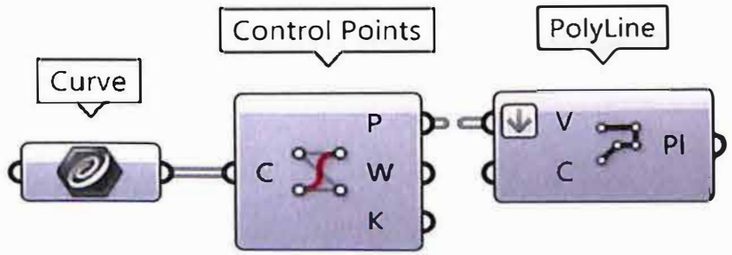


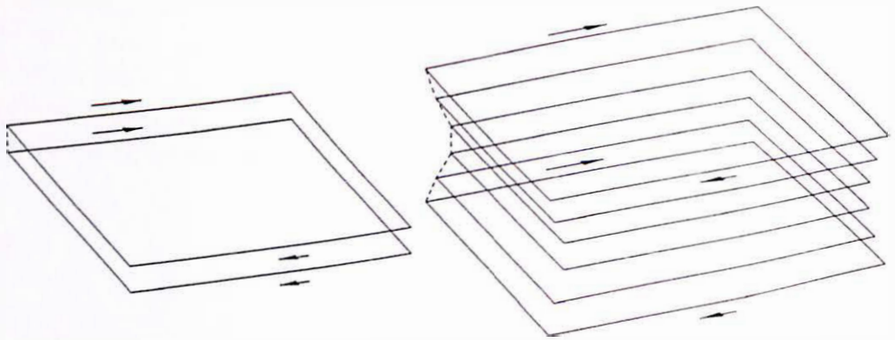
The values for the extruder are repeated. It works!

When the curves are not in XY plane, but moved in Z axis, the grammar we have created still works, but now, depending on the direction of the curves, the result could be better or worse.

In this example the previous definition works. But as the polylines have the start point in the same place, the printer will create a travel that could lead to loss of material between the end of the first curve and the start point of the next, and as such, not creating the best results. We can check the path, displaying it with a 'Polyline' component and flattening the list of points to create a continuous polyline.

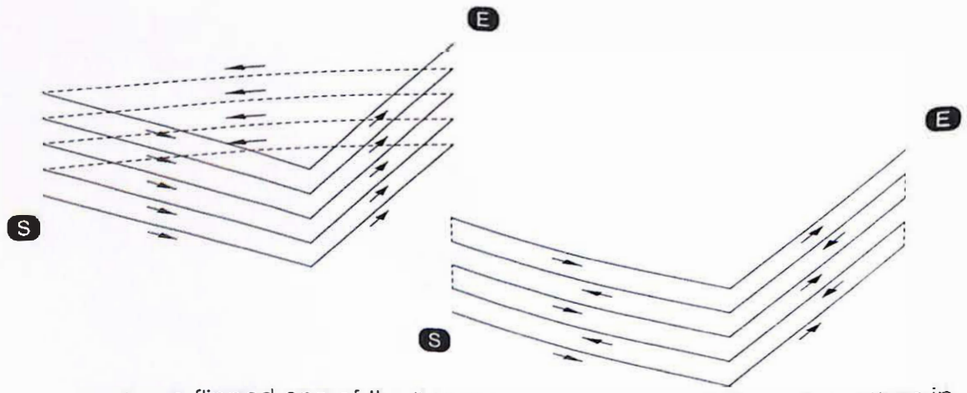






When the curves are closed and similar, like in the picture above, they do not present a problem as the seam points are very close one to the other.

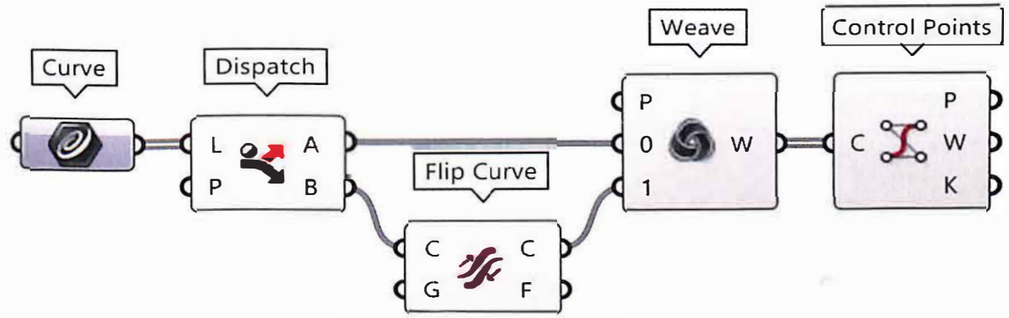
To solve the travelling problem in the example with two (or more) open polylines, it is as easy as flipping the direction of the second curve - or flipping the odd ones if there are more than two. For that, we must flip the direction of every other curve. It is possible to select every other polyline in one with the 'Dispatch' component and then flip the direction of the polylines of the output lists with 'Flip Curve' component. 'Dispatch' already has a pattern TRUE-FALSE in the 'Pattern' input. This means that the first curve on the list goes to 'A' output, the second to 'B', the third to 'A', and so on.



Once we have flipped one of the lists, it is necessary to put them together in the same order they had. 'Weave' works in the opposite way to 'Dispatch'. It puts together elements from different lists using a pattern. They are called 'streams'. By default the pattern input is '0,1' so the output list will be made of one element from list 0, the next from list 1, the next from list 0, and so on, so the

result will be the same list we had at the beginning with 'half' of the polylines flipped.

Notice the difference in the display of the paths. That system will improve the 3D printed object quality as well as printing duration.



The complex part comes in when we mix open and closed curves. There, it is up to the reader if it would be ok leaving it by default or changing the direction of the curves. This can be done in Grasshopper® but that means that the reader should be an advanced user to deal with data splitting the list, and then putting it together again.

Toolbar Menu

Analyze
Direction

Analyze
Main
Main2

For not so advanced users, there is a more 'physical' way to change the directions of the curves in Rhinoceros® with the 'Flip Direction' tool.

In this example we have only changed one curve to optimize the path.