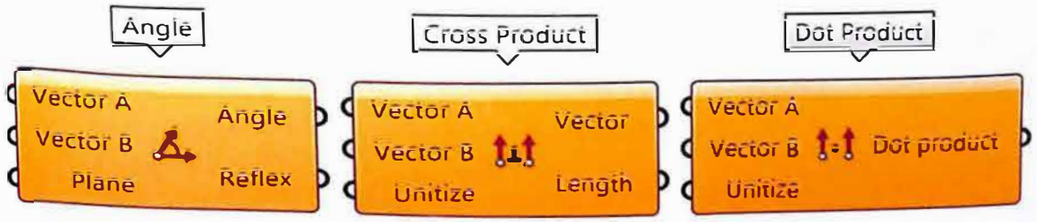
Vector

When the attractor is a vector.

Sometimes we want to use a vector that could represent the direction of a ray of sun light, a point of view, a wind direction, or the direction to a milestone in a city - such as a cathedral - whatever the attractor direction represents, we want our design to react to it. For that we need to compare the vectors of our project, in this example the normal vectors to the wave peaks with the attractor vectors. There are three simple tools in Grasshopper® to relate vectors:



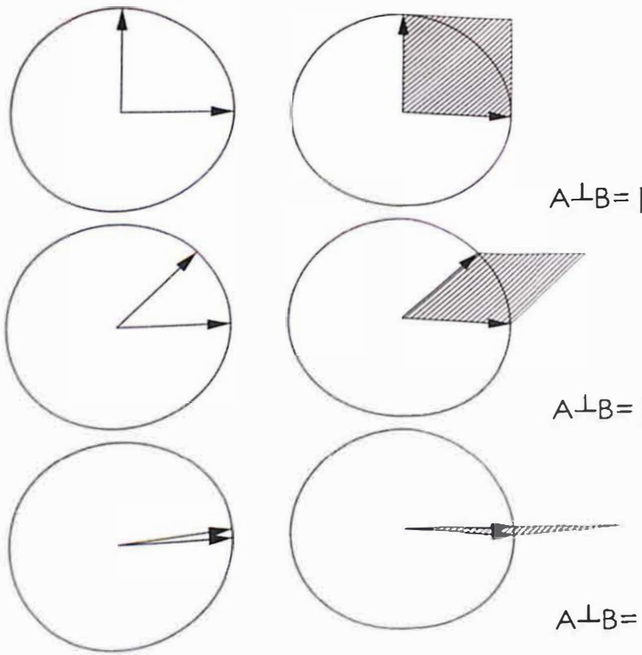
'Angle' (Computes the angle between two vectors). The output is an angle in radians.

'Cross Product' (compute vector cross product). The output is the normal vector to the plane formed by the original vectors ($\bar{A}$ and $\bar{B}$). It lies on the line of those vectors' resultant angle (β):

$$\bar{A} \perp \bar{B} = |\bar{A}| \times |\bar{B}| \times \sin\beta.$$

The cross product is well known by most, as the 'right-hand rule' describes its behaviour. It is also present every time we open the cap of a bottle that has a thread on it or when we screw a screw: turning in a clockwise direction means that the screw goes in, counter clockwise direction means that the screw goes out (resultant vector with same orientation, but opposite direction).

Geometrically it is useful to visualise the area of a parallelogram with one side representing one of the above vectors, and the other side being the other vector. E.g. with vectors of length 1 unit. Examples for 90°, 45° and 5°:



$A \perp B = |1| \times |1| \times \sin 90 = 1.00$

$A \perp B = |1| \times |1| \times \sin 45 = 0.71$

$A \perp B = |1| \times |1| \times \sin 5 = 0.09$

'Dot Product' (compute vector dot product). The output is a number. It is a relationship between the two vectors (A and B) that lies on the cosine of the vectors' resultant angle (β):

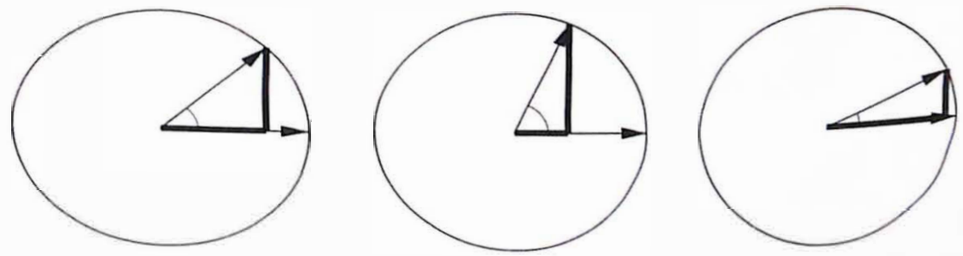
$A \cdot B = |A| \times |B| \times \cos\beta$

Before going on, let's remember what the sine and the cosine stand for in a right angled triangle:

The sine of an angle is the ratio between the opposite side and the hypotenuse.

The cosine of an angle is the ratio between the adjacent side and the hypotenuse.

Let's compare the angle of our two vectors of 1 unit length - say one of the vectors is the triangle's hypotenuse and is equal to 1. This means that the sine is the vertical projection or opposite side divided by 1, so it will be equal to the opposite side. The cosine is the horizontal projection or adjacent side divided by 1, so it will be equal to that side too:



The bigger the angle is, the sine will be larger and the cosine shorter.

The smaller the angle is, the sine will be shorter and the cosine larger.

This leads us again to the cross product and the dot product.

$A \perp B = |A| \times |B| \times \sin\beta$

$A \cdot B = |A| \times |B| \times \cos\beta$

For lengths of 1 unit vector, the result lies only on the angle ($|A| \times |B| = 1 \times 1 = 1$) and is opposite for the sine and the cosine. As Grasshopper® is aware of this idea, there is an input in both components to 'Unitize' the vectors which makes them 1 unit length no matter what their real length is, so the only difference is the angle. This makes things easier. It is more or less an equivalent concept to 'reparameterize' but for vectors so to speak.

As 'Cross Product' outputs a vector, let's focus on 'Dot Product' that outputs a number. If we 'Unitize' the vectors compared at 'Dot Product', then the result will be the cosine of the angle, a number between -1 and 1.

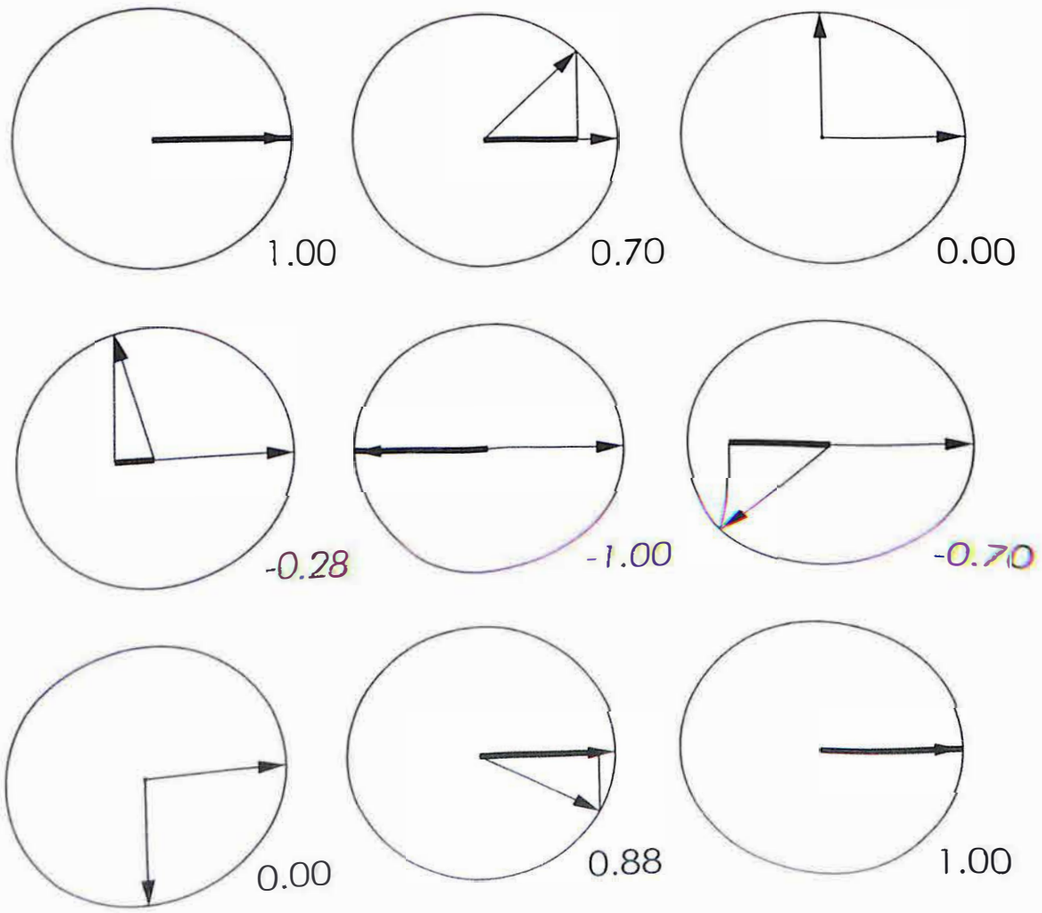
If the vectors are parallel with the same direction the 'Dot Product'=1.

If the vectors are parallel but opposite directions, 'Dot Product' = -1.

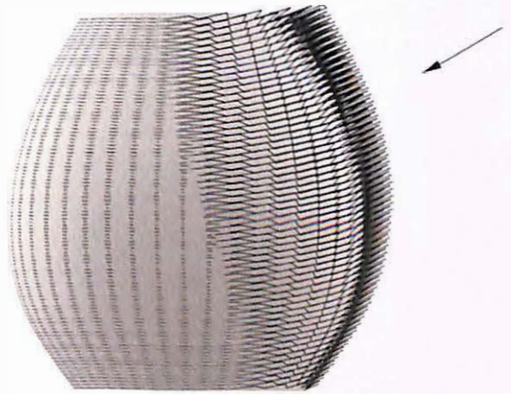
Other relationships in between will output intermediate values.

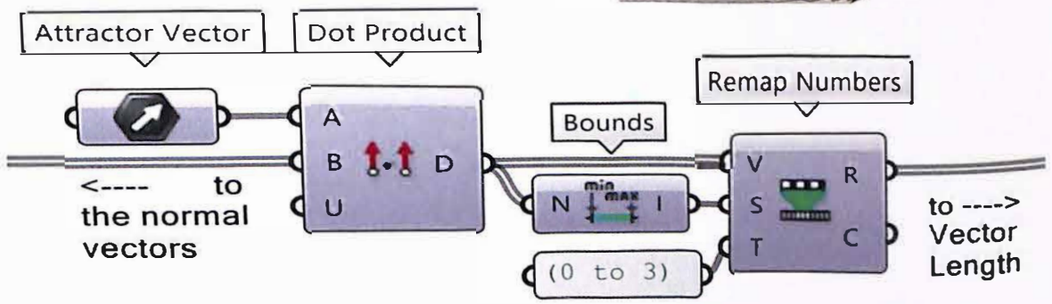
This logic is very useful. E.g.:

1. When we have a building and we want to open or close the window's blinders according to the amount of sun that enters the room.
2. To lighten or darken a colour according to the amount of light that lights the surface. This is the main tool that rendering engines use.
3. To perform-base wind design on an object.
4. And many others...

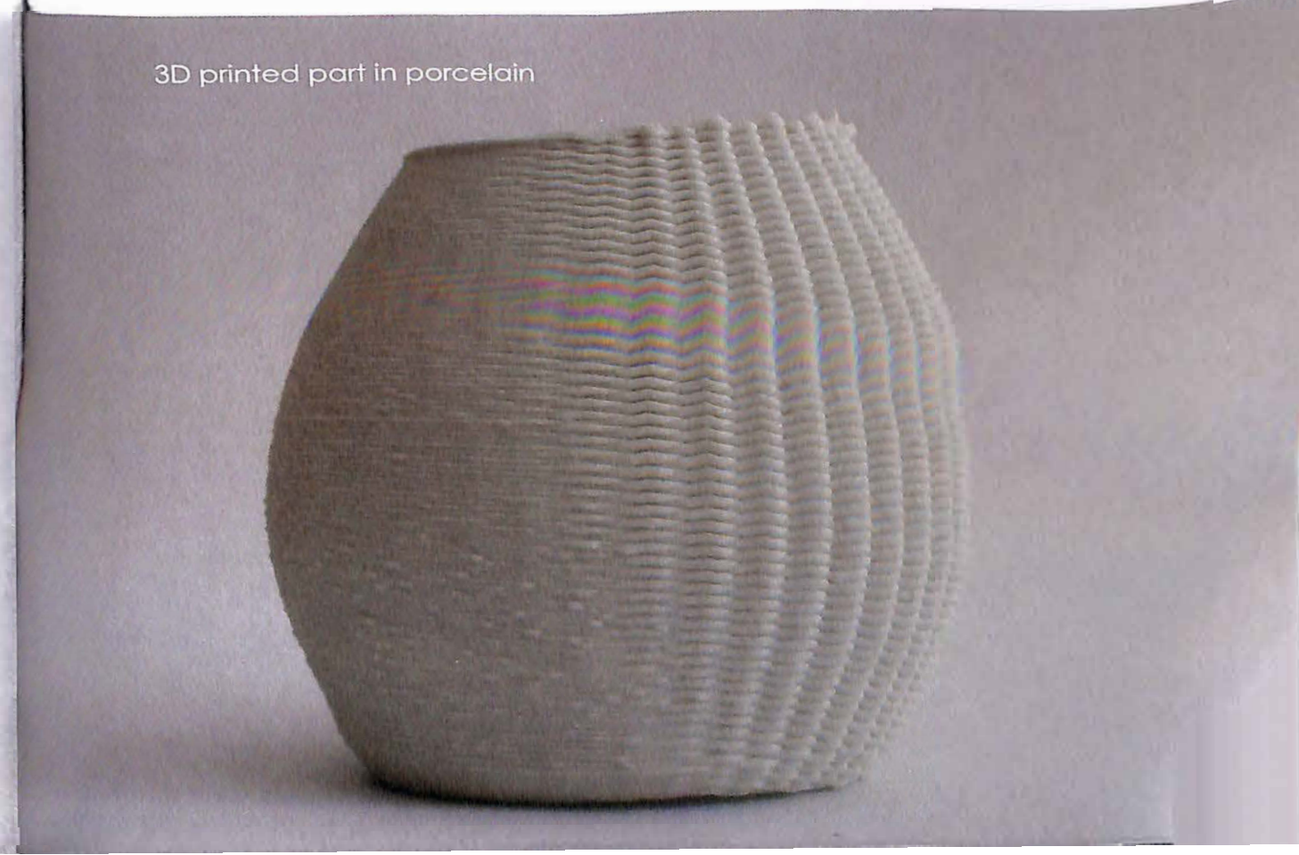


Now that we know how to deal with vectors, the values from 'Dot Product' should go as usual via 'Remap Numbers' and feed an 'Amplitude' component that controls the vectors' final lengths.





3D printed part in porcelain



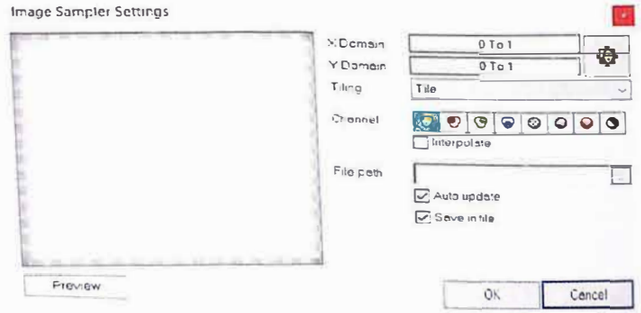
3D printed part in white PLA using a picture as attractor.

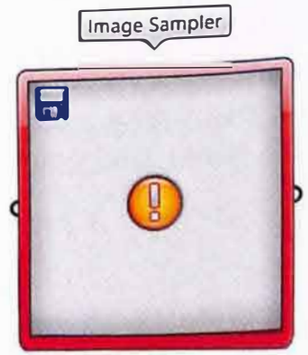
Waves pointing inside.

Image

Working with images is one of the simplest ways to modify a geometry.

The component to reference an image in Grasshopper® is 'Image Sampler'. This tool reads information from the pixels of the image and there are different parameters that we can extract from it. They are called filters or 'Channels': *RGBA colours, Red channel, Green channel, Blue channel, Alpha channel, Colour Hue, Colour saturation* and *Colour brightness*.





The channel *RGBA colours* outputs 3 coordinates between 0 and 255 which sometimes, especially talking about geometry, makes it difficult to control. The easiest output to work with is *Colour brightness* as it outputs a number between 0 and 1:

0 is the value for a black pixel **1** is the value for a white pixel

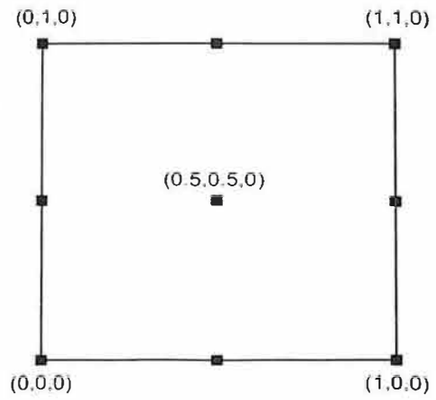
Grey colours are in between.

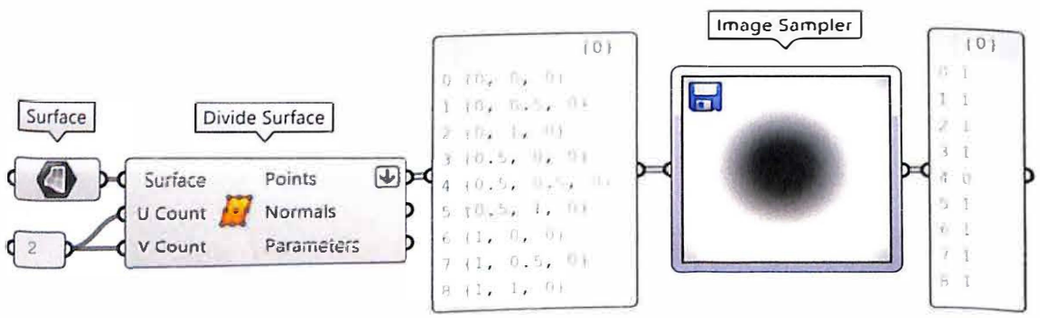
If the picture is in grey scale, those will be the final colours used. If it is full colour, the component transforms the picture to grey scale automatically.

Once we select the picture (right-click 'Image...') the component fits the picture into a grid that goes from 0 to 1 along the X and Y directions. It is possible to change this to any domain desired but working by default is fairly useful, as there are many ways to transform data easily from a given range to 0 to 1 through 'Reparameterize' or the component 'Remap numbers'.

'Image Sampler' only has one input. It has to be fed with a list of X and Y coordinates (it does not matter if there are Z coordinates as they will not be taken in account). Those coordinates enter the grid of the image and output the information of the exact pixel underneath. Depending on the type of filter selected, it could produce one or multiple numbers as explained before.

E.g. A concentric picture and a surface referenced in Rhinoceros®, with a size from 0 to 1, equivalent to the picture's domain. 'Divide Surface' is used to extract points that enter the picture and output points on the surface. The filter selected is 'Colour Brightness' so a black pixel like the one present at 0.5,0.5,0 coordinate above will output a value of 0 whereas a white one, like the rest of them, will output 1.





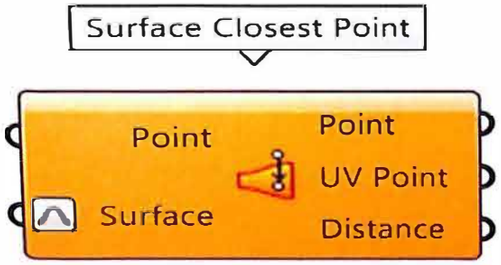
'Divide Surface' is a component that can find points on a surface. It can be used as an alternative to the 'Contour' strategy explained at chapter *Brep -> Curves -> Polyline -> Points -> X,Y,Z*. But this component only works with surfaces, not with polysurfaces or extrusions. It has three outputs:

'Points': physical points with their coordinates related to the cartesian coordinate system with origin 0,0,0.

'Normals': the normal (perpendicular) vectors to the surface at the division points.

'Parameters': parameters or 'local' coordinates according to the surface domain. The domain of the surface is not always that easy to figure out. For that reason, Grasshopper® ® has the option to reparameterize objects as surfaces. This means that the surface domain will go from 0 to 1 in both directions U and V, no matter the size or the position of the surface in the cartesian space. So if we 'Reparameterize' the surface, it will have the same domains as the picture. Use the output 'Parameters' as (X,Y) coordinates to enter the image and extract information of blacks and whites.

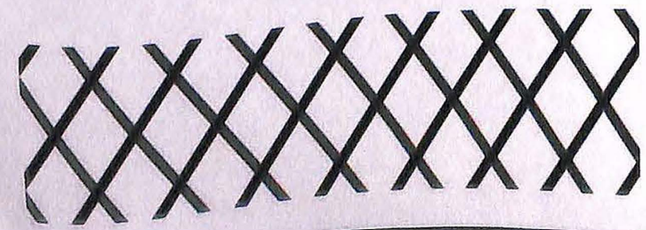
If we have followed the steps at chapter *Brep -> Curves -> Polyline -> Points -> X,Y,Z*, and the Brep is a surface, we can get U,V parameters from the points of the polylines to feed an 'Image Sampler'. Use 'Surface Closest Points' to get the U,V Points from the vertexes of the polylines. Remember to 'reprameterize' the surface.

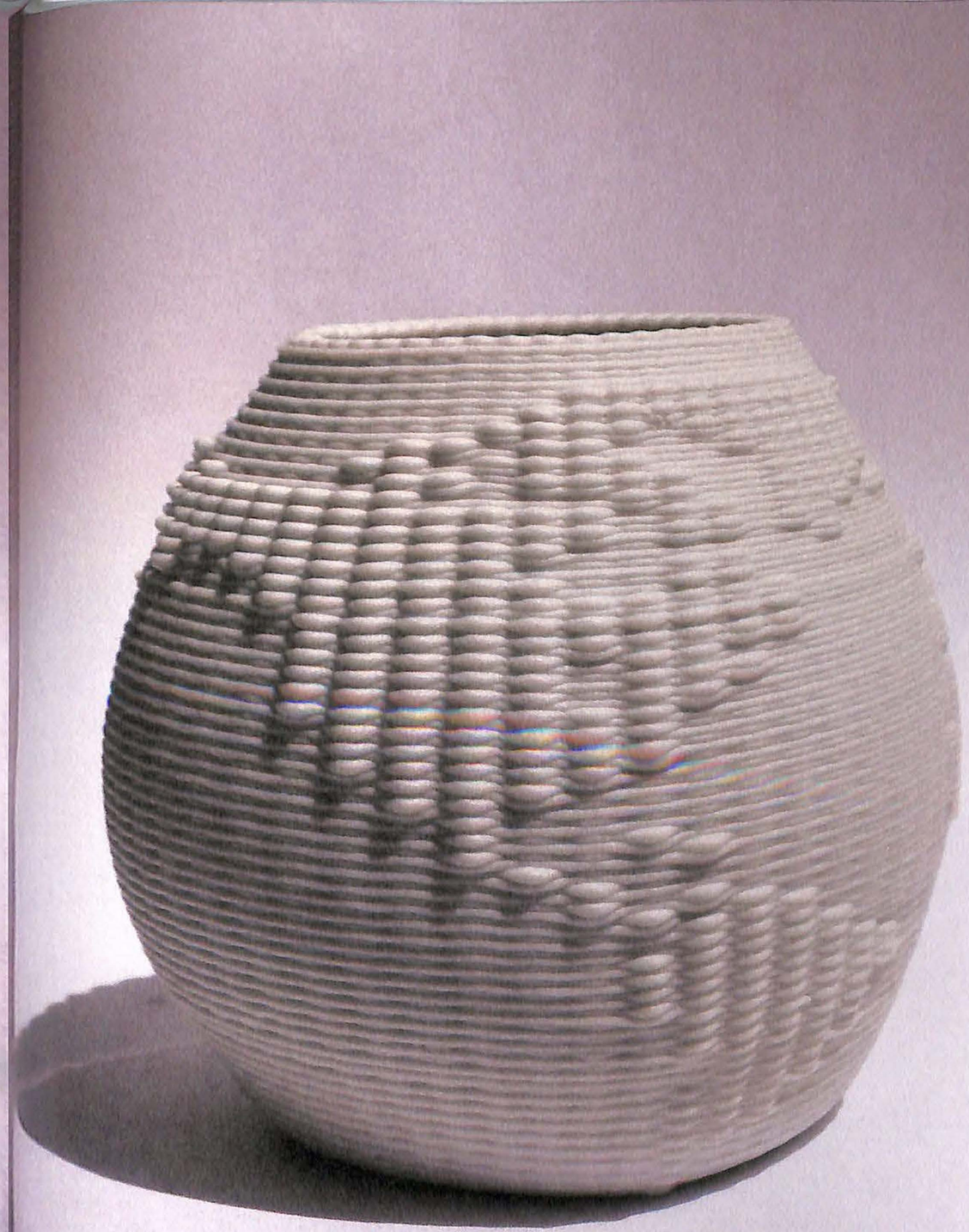


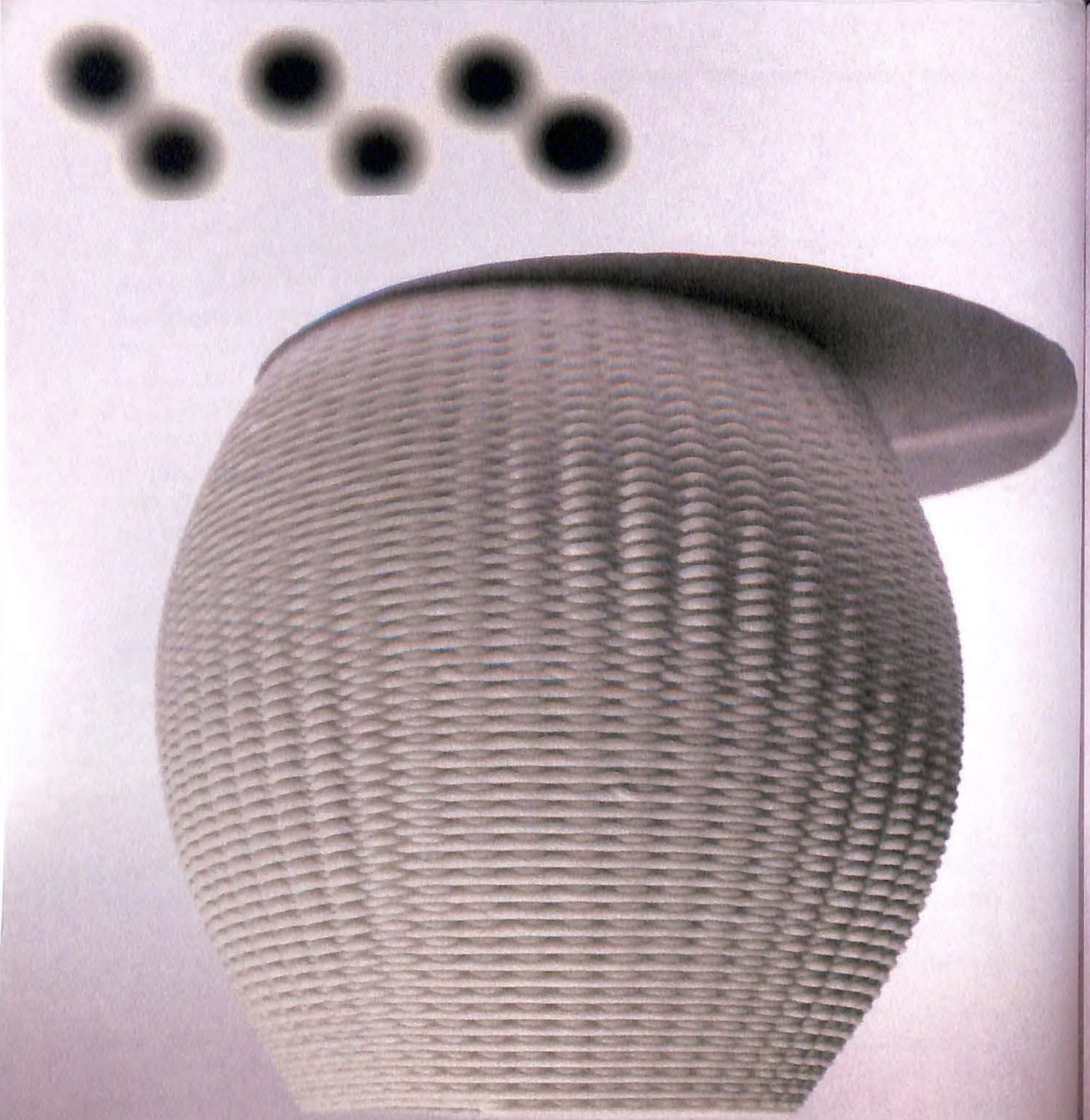
Once we have the data from the picture, as usual, it is up to us on how we wish to use it. Following previous examples, it has been used to change the vector lengths for the waves' peaks, but other parameters could also be easily varied, such as the speed, the flow, etc.

Examples in the next pages.

3D printed parts in porcelain with corresponding reference images.







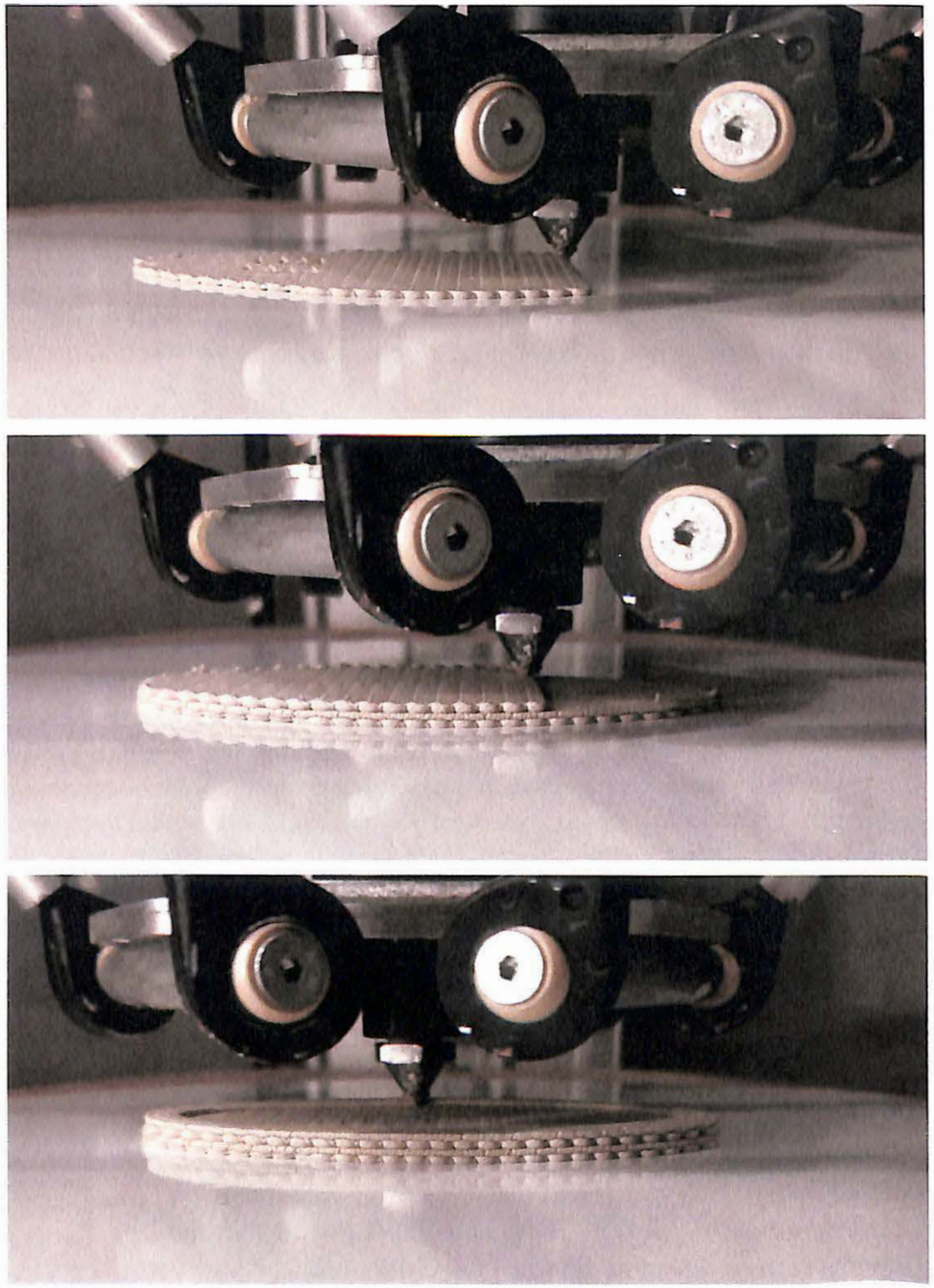
Feed rate (F)

Also known as *speed*. Being able to change the speed of the machine depending on some parameters could provide interesting results, particularly when working with a 3d printer with a constant extrusion and with no possibility of retraction like some clay 3d printers. It could be an interesting alternative to flow edition.

In the g-code we have developed, the speed is constant (e.g. 1800 mm/m.), and the flow is evaluated according to the distance between the points or the lengths of the polyline segments. If we want to display the effects of a changing speed, we must do the opposite: fix the flow (E) value and modify the speed (F). This means that the extrusion is going to be constant.

Modifying the F value on the g-code can be done via a series of data such as numbers that multiply the F value. But it can also be done in a more creative way by using the image strategy from the previous chapter. In order to achieve better results and display the image on the 3d printed object, the distance between points has to be the same. Then, as in the previous chapter, get the 'Parameters' of the points on the surface, extract data from 0 (black) to 1 (white) from 'Image Sampler' and use those numbers as a percentage to multiply the speed for each segment between points.

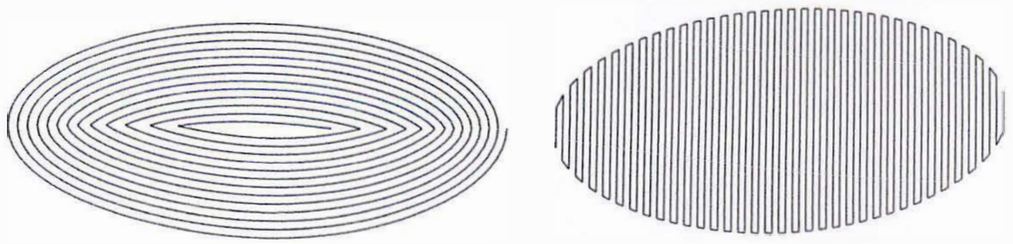
Fixing the Flow (E) is only possible if our machine allows us to separate the extrusion from the movement. For example, an 'unconnected' printer clay extruder activated by a 1|0 digital input would be easy to control with a constant extrusion. But if it is a standard 3d printer, we cannot split the movement from the extrusion as the extruder behaves like an extra axis. Every line of G1 g-code needs a value for E. The solution works via providing variable proportional values for E and F depending on the image's blacks and whites – if we are using an image. So, it will be a combination of E and F values. That percentage should be inversely proportional for E against F. This means that the flow (E) should become smaller while the speed (F) becomes faster and vice versa.



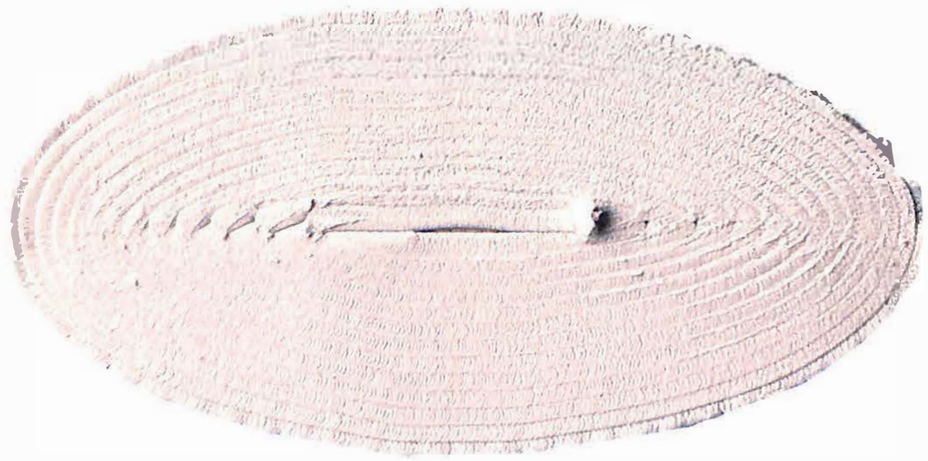
WASP 4020 3D printer. Porcelain

Base

The base of our design could easily be done by hand if the material is clay, or it could be designed in Grasshopper®. Some owners of a clay 3d printer prefer to make it by hand as it is much faster. If we are working with thermoplastic or if we prefer to 3d print it, making a solid base could become interesting by combining two strategies: 'concentric curve' and 'contours'. The combination of both will make a stronger base.



Both are variations of two previously explained strategies. In both we start with a curve, in this example, an ellipse of 80 mm by 40 mm that defines the perimeter. For other curves or geometries, we may have to modify the following definitions.



Concentric

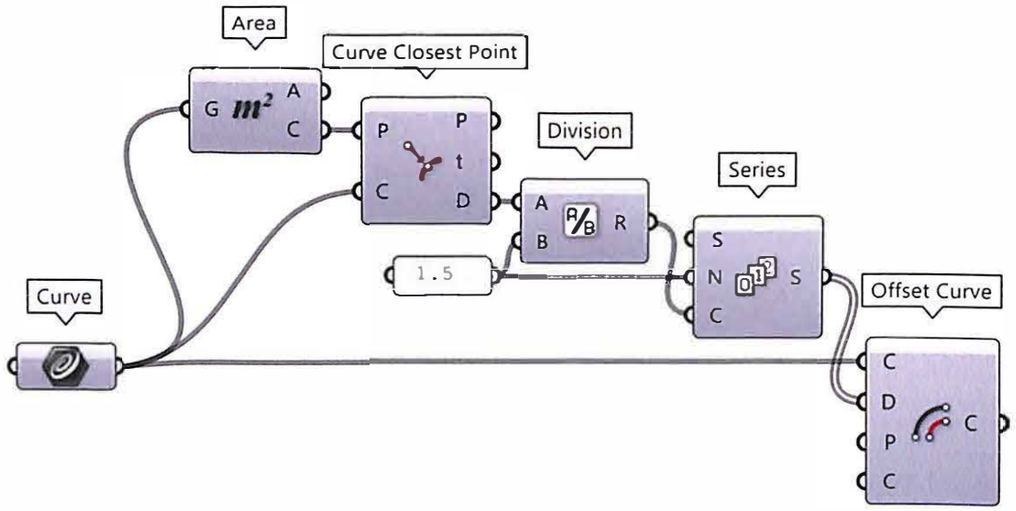
Depending on the design flow, the ellipse could be drawn in Rhinoceros®, in Grasshopper®, or we could extract it from the 3D object with tools such as '_DupBorder' in Rhinoceros®, 'Brep Edges' in Grasshopper® or others. Let us get into the details of each strategy.

Use 'Offset Curve' to create interior offsets that infill the area.

Divide them into points and create vectors with 'Flip Matrix' to move the points gradually toward the correlating point on the next curve, as in the example in chapter *Brep -> Curves -> Polyline -> Points -> X,Y,Z*

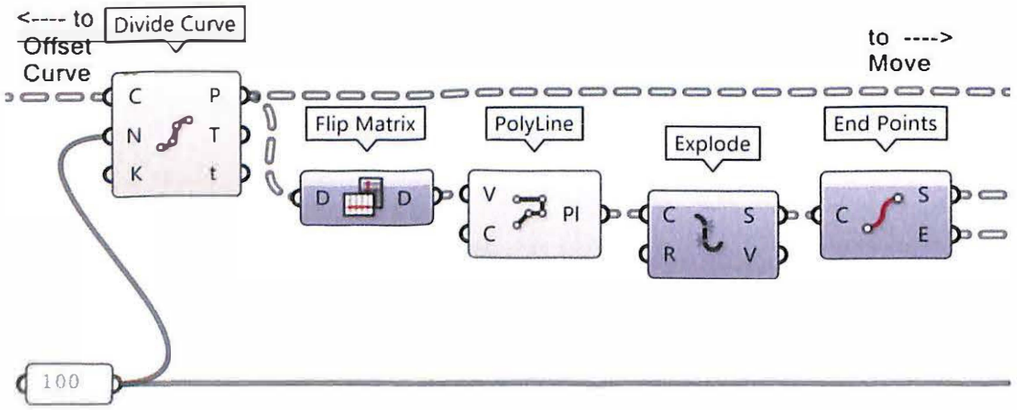
First create convenient offsets depending on the diameter of our nozzle. The base curve could be quite irregular, not just an ellipse, so depending on the geometry, we may face some problems. The ellipse has two different distances to the centre so first decision to take is deciding which is going to be our reference distance. If we decide to use the shorter distance, we can measure this with the tool 'Curve Closest Point'. If we prefer the maximum distance, we can use the tool 'Extremes' in combination with a proper plane.

The 'Area' component outputs the centroid of the curve. In combination with 'Curve Closest Point' we can get the shortest distance and with a 'Series' component, create a series of values for 'Offset Curve'. In the example below, the 'Curve' is an ellipse from Rhinoceros® and the diameter of the nozzle is set to 1.5 mm.

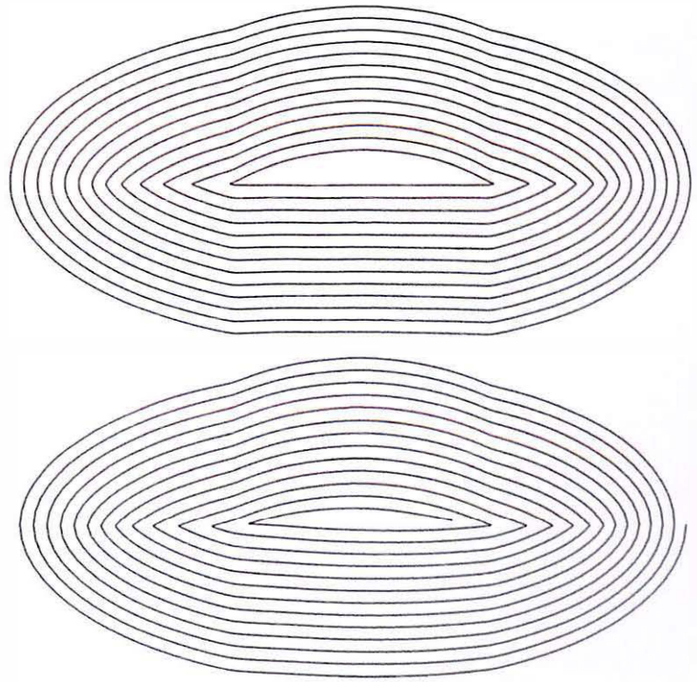


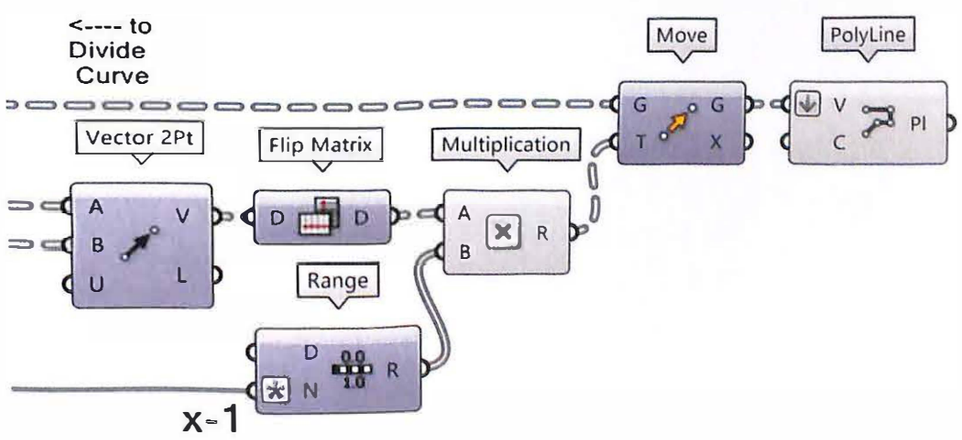
The rest of the grammar is as explained at chapter *Brep -> Curves -> Polyline -> Points -> X,Y,Z*. We will summarise this again, but it is recommended to review it if there is any doubt.

Use 'Divide Curve' to get division points. Create vectors with 'Vector 2pt' that go from one point to the correlating one in the next offset curve. As the data tree responds to the offset curves and not to the vector directions, we need to use 'Flip Matrix' before constructing the polylines and after the vectors to go back to the original data tree structure.



'Range' divides a domain from 0 to 1 to create a percentage that will modify the size of the vectors. In the 'Steps' input there is an 'Expression', x-1, because 'Range' creates one extra value (101 instead of 100) that must be subtracted.

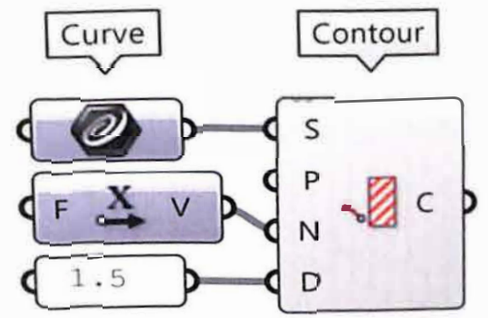




Contours

The 'contours' strategy is similar to the one explained at chapter *Mesh -> Polylines -> Points -> X,Y,Z*

In this example with an ellipse, as it is a closed curve, when connected to 'Contour' the software considers it as a 'Shape' and generates the sections according to the parameters given for distance (the nozzle diameter, in our example, 1.5 mm.) and direction (X):

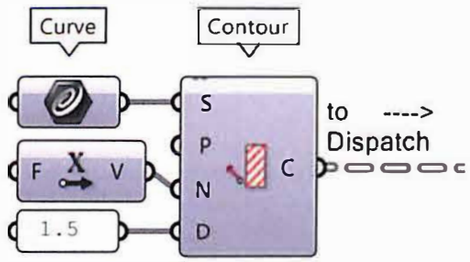


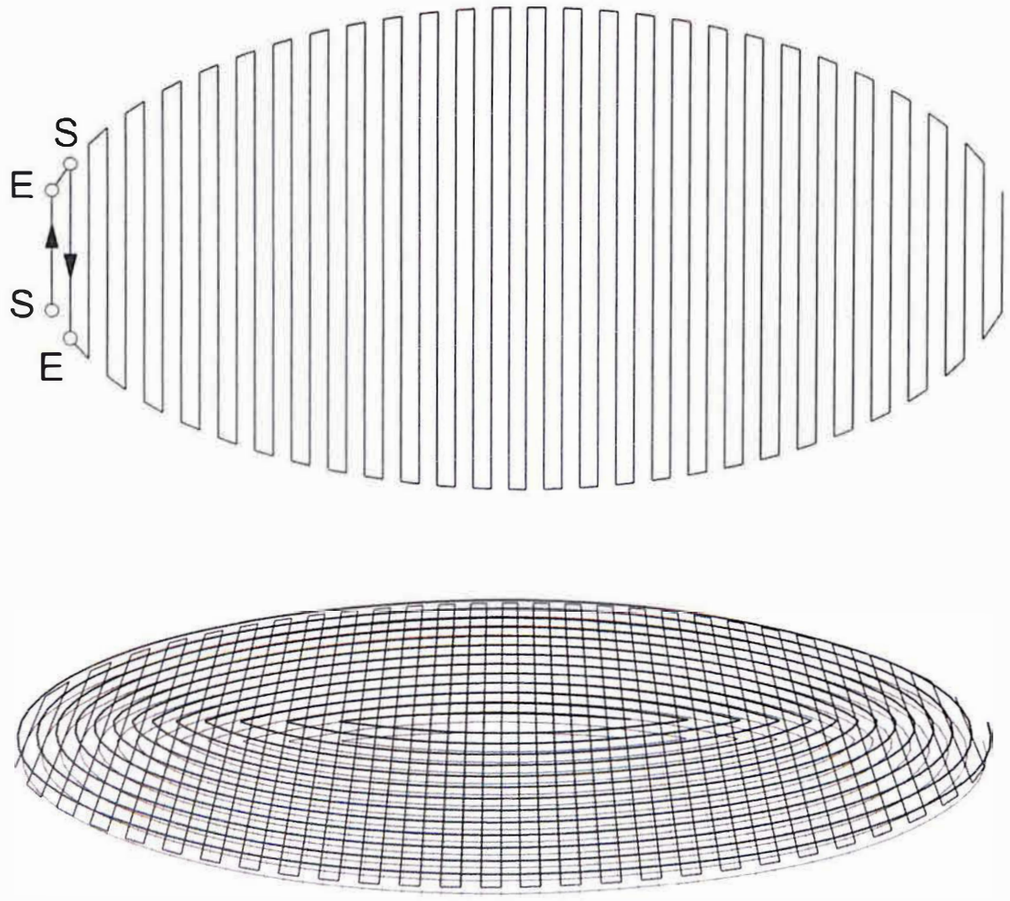
As these curves have the same direction, when 3D printing, we will create a lot of unnecessary travels from the end point of one to the start point of the next. As explained at chapter Polylines -> Points -> X,Y,Z we can flip the direction of every other curve so that the start points of the even curves will be the closest points to the end point of the add ones.

The solution is achieved via splitting the list of curves with 'Dispatch' into two lists and flipping the direction of the curves in one of the lists.

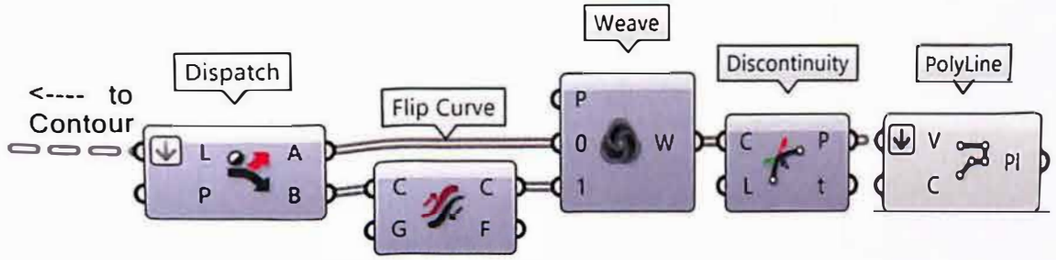
As 'Dispatch' splits the original list into two new lists, after 'Flip Curve' we need to restore the curves to their previous structure. Use 'Weave' to intertwine the curves from both lists, first one from 'List A' and then one form 'Flip Curve'.

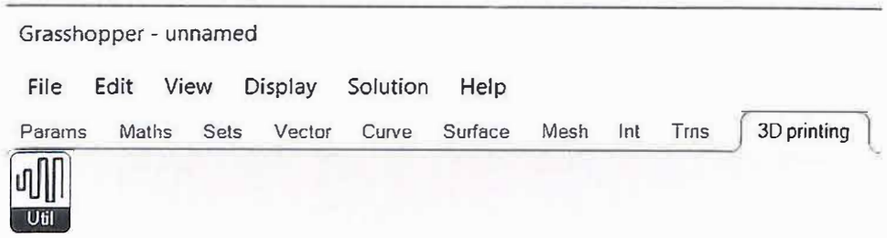
To create a continuous extrusion, we need to extract the start and the end points and draw a polyline. Use 'Discontinuity' to find the end points and feed the 'Polyline' component with them. It is necessary to 'Flatten' the data, as 'Discontinuity' will group the points in lists, as many lists as curves, with two points (start and end) per list.





The two strategies could be combined to create a thicker and more solid base



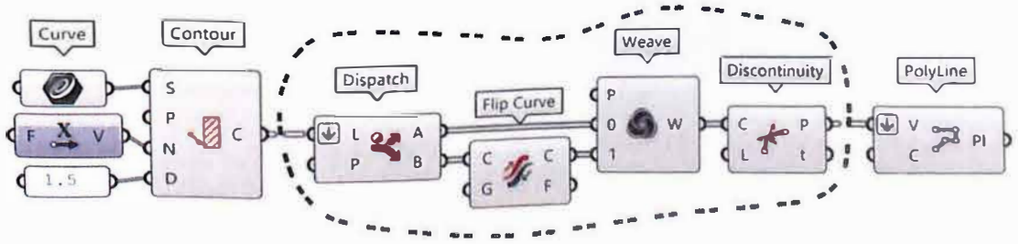


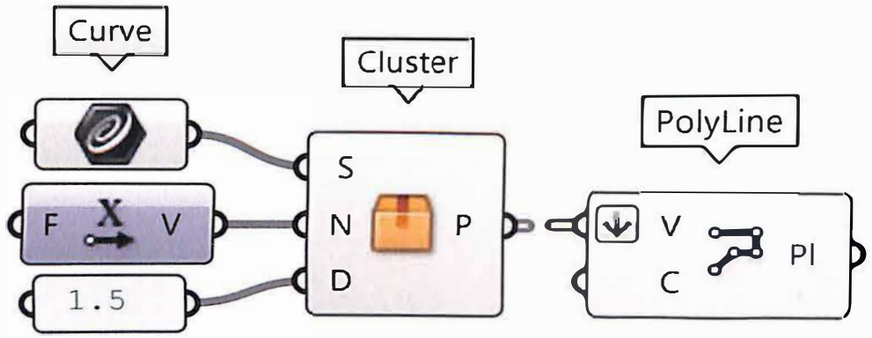
New tab with submenu in ribbon, for your custom tools

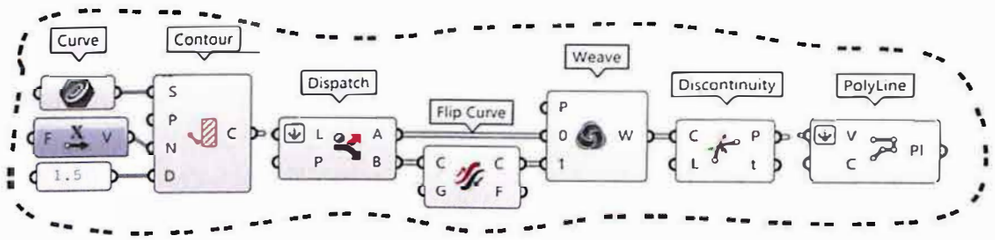
Clusters

'Cluster' is an option in Grasshopper® that allows us to pack several components into one single tool. It is a powerful and useful way to organize your definitions. It can be used on small parts of our definition to simplify the processes involved or could be used for the entire grammar to create our own tools (which could become very rewarding) or new tabs similar to plug-ins known as 'User Objects'. Of course, such clusters will not become official new components, as for that we need to work with SDK documentation and code in Visual Basic or C#, but it is a simple and nice approach to that 'world' for standard Grasshopper® users.

A 'Cluster' is like a Grasshopper® document inside a Grasshopper® document. To create one, select the components to be clustered, right click or middle button and left click on 'Cluster'. If we select certain components inside a definition, those selected components will be grouped in a new tool with as many inputs and outputs as we originally had connecting to the perimeter of the new group. Let's see an example with the last definition at chapter Base/Contour:



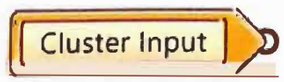


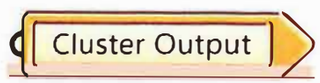


If we cluster all the components in the grammar, the new tool will be a 'Cluster' without inputs nor outputs:

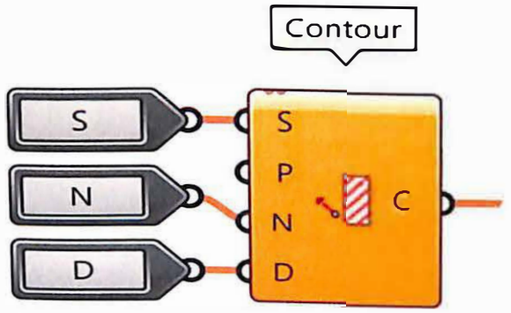
Double click on the 'Cluster' and the grouped components will pop up in a new window where they can then be edited.

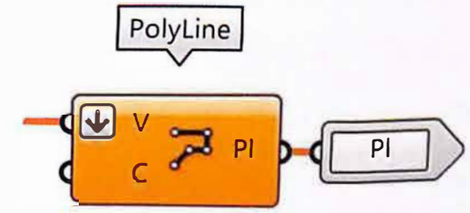
In the first example the inputs and outputs are automatically created using the following components that can be found at the 'Param' menu tab in the ribbon menu. For the second, we need to add them manually, choosing which elements we want to become outputs and inputs in our 'Cluster':





Use 'Cluster Input' for 'Shape(S)', 'Direction(N)' and 'Distance(D)' and use 'Cluster Output' for 'Polyline(Pl)'. Then 'Cluster' the set of components. We have

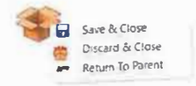




created a new cluster as a tool that outputs a polyline depending on three inputs, a curve, a vector and a number that represents the nozzle diameter:

The 'Cluster' adopts the names of the inputs and outputs of 'Contour' and 'Polyline'. To change them, right click on each one and type the new names: 'Shape', 'Normal Direction' and 'Nozzle Diameter' for the inputs and 'Polyline' for the output (it does not have to be these exact names specified, these are just a suitable example as other names will also work).

Then right click over the icon to explore several interesting options:



Edit Cluster...: It is equivalent to double click. A new window will pop up where we can edit the content. There are options available to save the edition, or not.

Properties: Add a name, a nickname, a description and change the icon. The name is the complete name for the tag. The nickname is the name that appears when we don't display icons. The icon can be chosen from the list of icons that Grasshopper® has already available or it can be a custom image. If you chose the latter option, it is recommended to make it in .png format with transparent background.

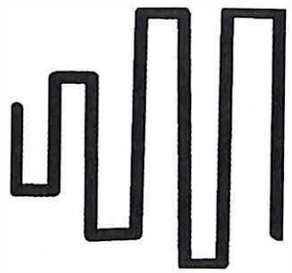
For our example:

Name: 'Base Contours'.

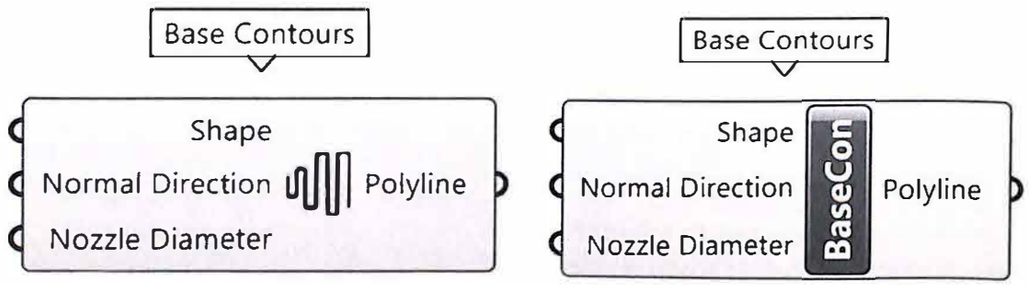
Nickname: 'BaseCon'.

Description: Creates a continuous polyline for a Base.

Icon: draw it and save it as .png



The final Cluster will have this appearance with and without icons display:



Assign Password. A password can be assigned to our cluster. It means that the cluster can be used, but not opened or modified without the password. This is very useful if you want to keep the 'secret' of your magic or if you don't want another user to 'destroy' your masterpiece by editing its content. For example, if you want your students to edit inputs and explore different outputs, but not to modify the inside of the cluster. Either way, it seems that the addition of a password does not make your cluster 100% safe, as Grasshopper® needs to directly work with the cluster when a file is open, and an advanced programmer could steal the cluster information.

Explode Cluster. This option will undo the 'Cluster'.

Disentangle. Every 'Cluster' has an ID which is created randomly with the 'Cluster'. Copied 'Clusters' maintain their assigned ID. 'Clusters' with the same ID are modified simultaneously when one of them is edited. We could say that they behave like a 'block' instance. So, if one 'Cluster' is edited, each copy will be edited simultaneously unless we uncheck the 'Disentangle' option.

Export. Creates a new *.ghcluster file based on that 'Cluster'. For example, to be used in another *.gh file.

Export & Reference. Similar to 'Export', but it creates a link between the file in which the original 'Cluster' was created and any files where it is then used in afterwards. So, if the 'Cluster' is changed in the original file, it will also change in the second file. This is a similar behaviour to a 'block' instance between files.

Update. It is useful to regularly update a 'Cluster' in a file when it has been edited in the original file and was exported using 'Export & Reference'.

Internalise. This option is only available when we are using a referenced 'Cluster'. Selecting it, the current 'Cluster' will be dereferenced from the original source.

User object.

The 'User Object' option is a way to create your own tabs in the ribbon menu. This is very useful when used in combination with 'Clusters' as that way you have them both ready to use in your Grasshopper® session.

It is as simple as selecting one (only one at a time) 'Cluster', 'File', 'Create User Object...'. The category will be the name of the tag and the Sub-Category the sub menu.

This is a more similar and easy way to create something that looks like your 'own plug-in' – see picture at page 184. That would be the appearance of our ribbon with a 'User Object' named '3D Printing', with only one Sub-Category (Util) and our cluster 'Base Con'.

Important to note:

A new Grasshopper® file or Rhinoceros® session in the same computer will maintain 'Clusters' and 'User Objects' as they were created. When dragging and dropping a custom 'Cluster' from a file to another file or to another computer, the properties such as name or icon will be lost, and they should be updated manually (at least in the current version 1.0.0007). Anyway, these are very useful options.

3D printing experiment without supports. White PLA.

Wireframe

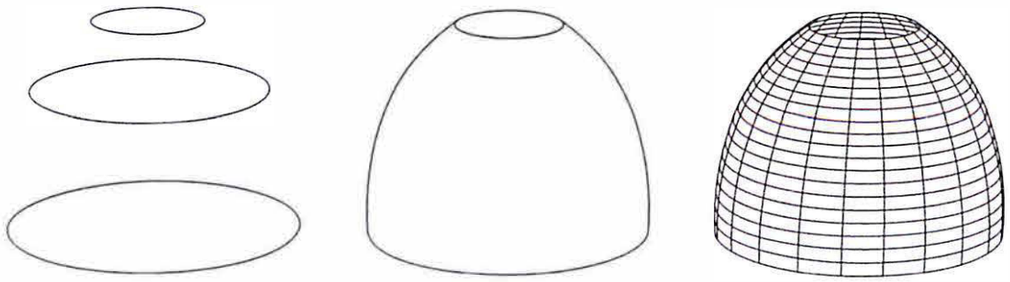
It is possible to 3D print structures like trusses in the air without supports using thermo filament. With clay it would be very difficul. It is the printer fan the one that has to cool down very fast the plastic in its position.

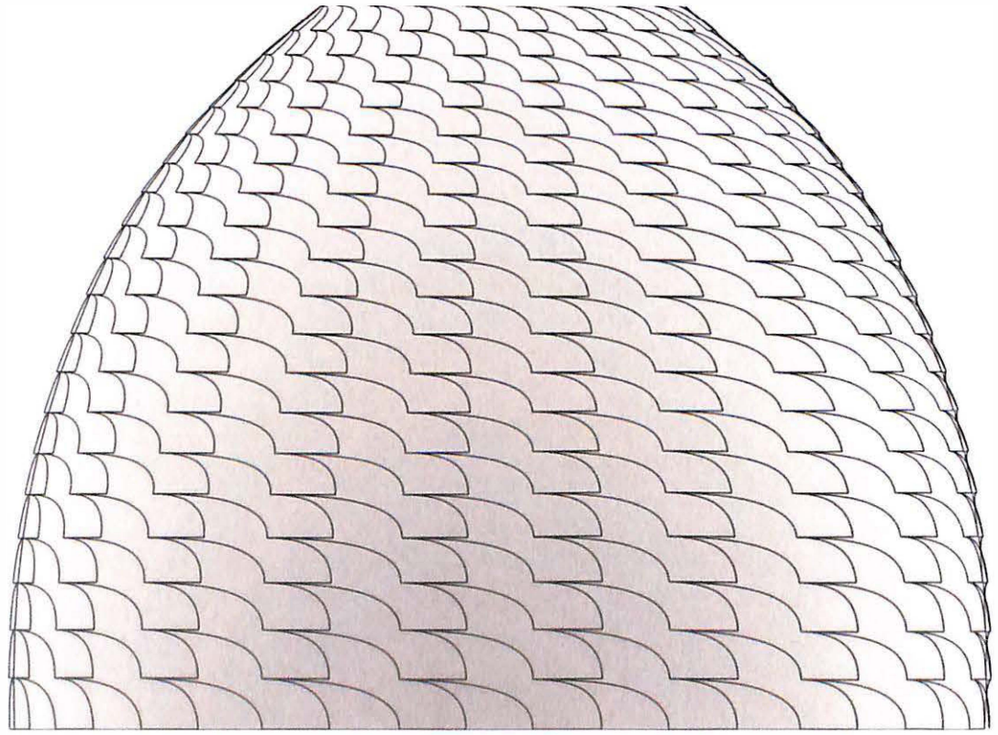
A Grasshopper® advanced user will easily find a linear structure on a surface the same way we have created curves in previous chapters with 'Contour' or 'IsoCurves'.

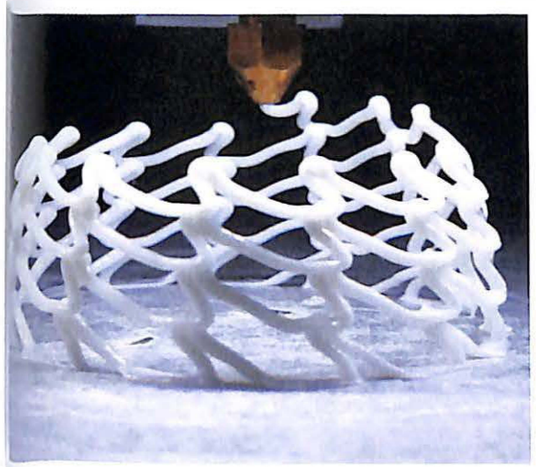
We suggest using subdividion domain tools as 'Divide Domain2'. The main part is defining both, direction and order for the curves with tools such as 'Flip' and 'Sort List' already explained. As usual, the geometry will be crucial. It will depend on the readers ability to find solution to the new vast range of possibilities.

In this example, we create a surface with a simple loft using three circles.

'Divide Domain2' + 'Isotrim' divide the surface in equal parts. With selection tools such a 'Deconstruct Brep' + 'List Item' and 'Point On Curve' it is possible to select subsurfaces' corners and mid points to draw the 3D path.



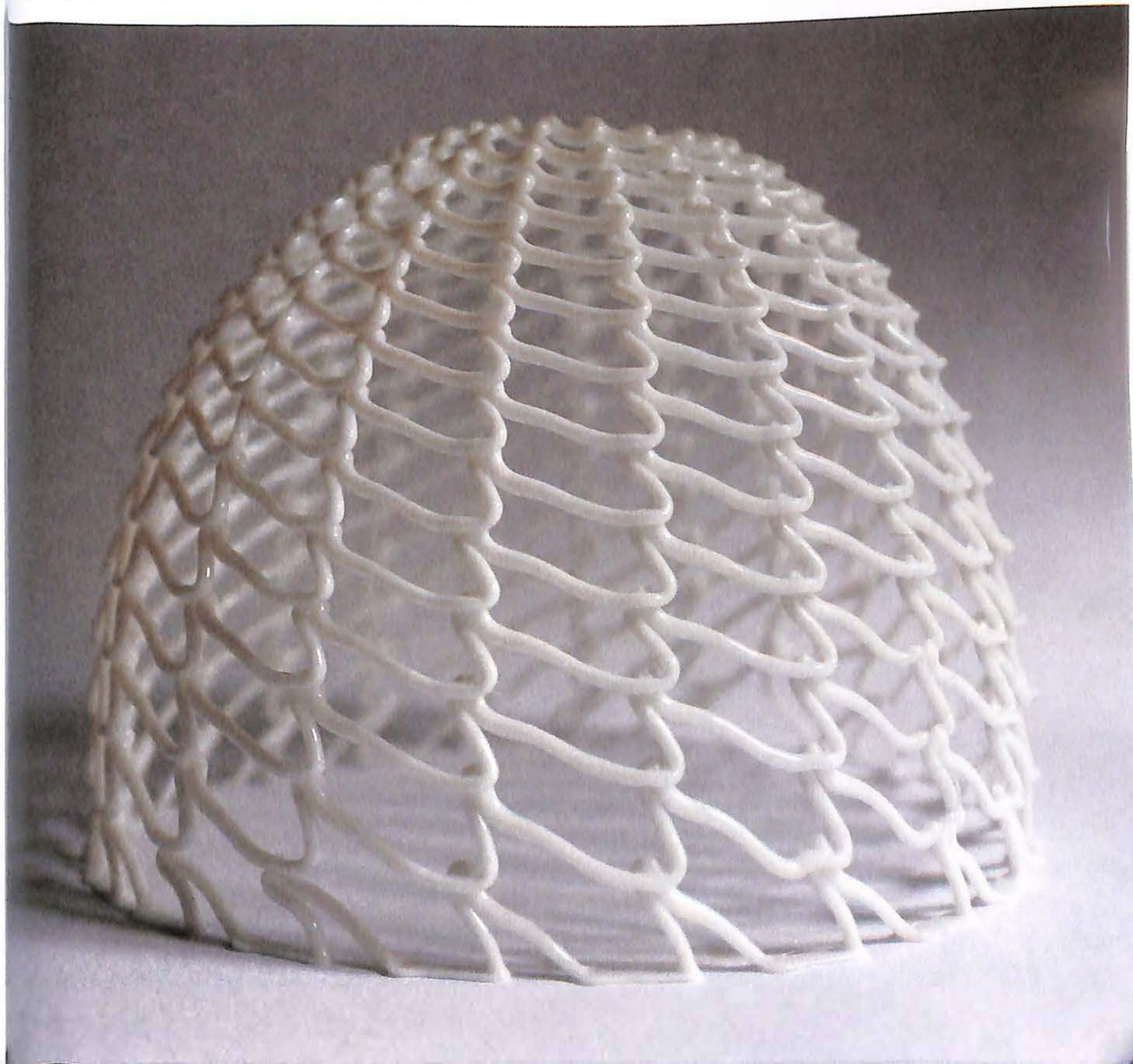




G4 is a G-code instruction that allows to stop or dwell the machine at a certain point. It can be used to stop the 3D printer and allow the plastic to harden.

Remember that every instruction for the G-code must be included in the 3d printer's firmware.

Enjoy!



Bibliography

Armstrong, Kate; et al. Design, Remix, Shape, Repeat: How distributed design is changing the way makers and designers approach collaboration, tools and the market. IAAC, 2019.

Di Marco Giancarlo. Simplified Complexity. Method for Advanced NURBS Modeling with Rhinoceros. Le Penseur, 2018.

EVOLO 06: digital and parametric architecture. Los Angeles, 2014.

García, César; el al. Deconstruyendo el manifiesto maker. Transit projectes. 2020.

Glendinnig, Paul. Math in minutes. Quercus, London, 2013.

Issa, Rajaa. The essential mathemathics for computational design. Robert Mcneel & Associates. 4th edition, 2019.

Marsden, J.E.; Tromba, A.J. Vector Calculus. W. H. Freeman, NY (1976); Fifth Edition, 2004.

Mattison Steve. Guía completa del Ceramista. Herramientas, materiales y técnicas. Ed. Blume, 2010

Menichinelli, Massimo. Fab Lab e maker: Laboratori, progettisti, comunità e imprese in Italia. Quodlibet Studio. Design. 2016.

Prusinkiewicz, P. The algorithmic beauty of plants (The virtual laboratory). Springer, 1996.

Rael, Ronald; San Fratello, Virginia. Printing Architecture: Innovative Recipes for 3D Printing. Chronicle Books, 2018.

Redwood, B; Schöffer, F.; Garret, B. The 3D Printing Handbook: Technologies, design and applications. 3D Hubs. Amsterdam, 2017.

RerRap project: https://reprap.org/wiki/RepRap

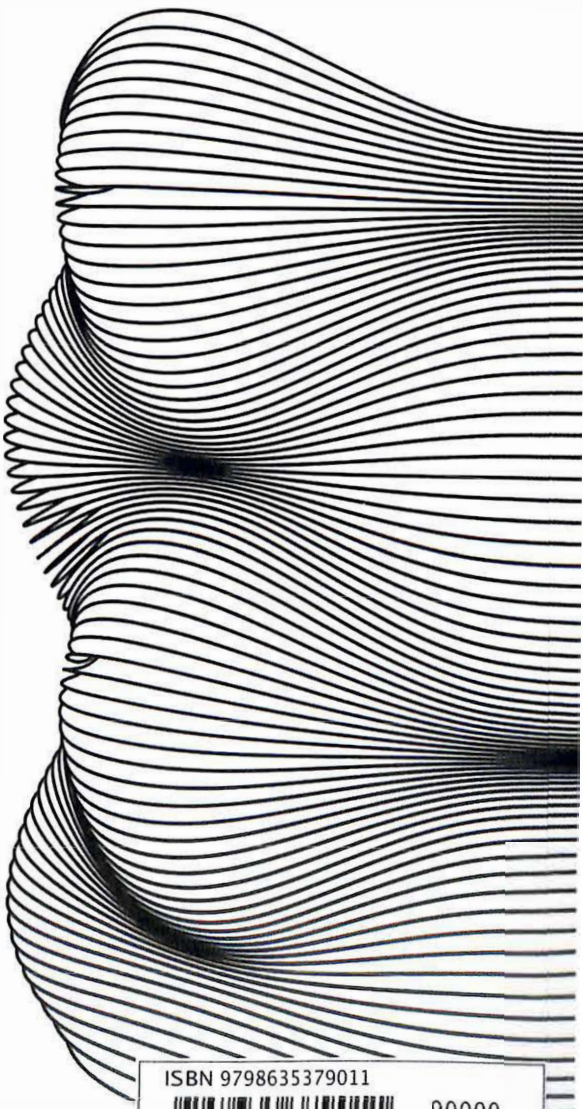
Tedeschi, Arturo. AAD Algorithms-Aided Design. Parametric strategies using Grasshopper. Le Penseur, Milano, 2014.

Thompson, Darcy. On Growth and Form. Cambridge.: Cambridge University Press, 1917.

West, Mark. The fabric formwork book: methods for building new architectural and structural forms in concrete. Routledge. NY, 2017

This book forms a connection between both the worlds of Grasshopper® and 3D printing, explaining how to transform a design into a series of curves and paths for a 3D printer to use. It teaches how to write and **create G-code** directly within Grasshopper®, **without the use of scripts or plug-ins**, in easy format for existing Grasshopper® users to understand. Big experience using the software is not required, but skilled users will be able to take full advantage of the content.
The book focused mainly on **clay 3D printing**, but the same logic can be applied to thermo-filament 3D printing (**FDM**) as well. The methods taught open up a wide range of new possibilities when 3D printing, like **non-planar** printing or non-conventional paths for the 3D printer.

Diego García Cuevas and Gianluca Pugliese are experts in 3D design and digital fabrication as well as teachers at several universities



ISBN 9798635379011