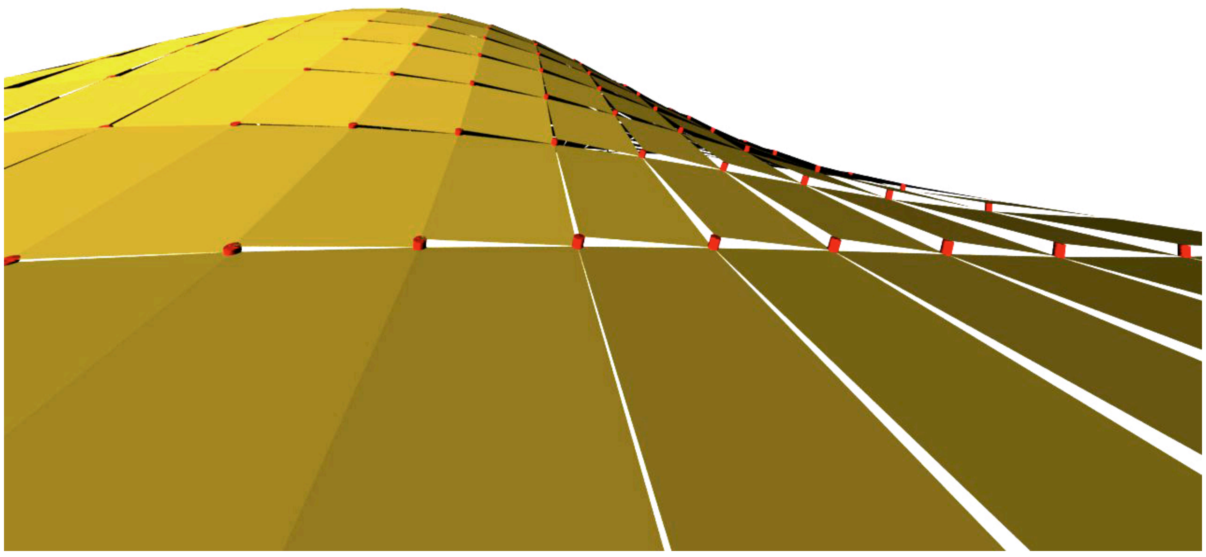
FREEFORM SURFACES WITH PLANAR QUADRILATERAL FACES



If you try to cover a freeform surface with quadrilateral faces, in general the quadrilateral faces are not planar, because their four vertices does not necessarily lie in the same plane.
However, having planar quadrilateral faces in architecture is often convenient because the surface can be fabricated conveniently from flat sheets of material (glass, steel, etc...).
The method proposed here is the application of vector operations to determine the geometry of planar faces covering any freeform surface.

What you will learn

1. The use of vector operations for application in advanced geometry
2. To manage and transform large sets of data through the use of indexes.

List of relevant components used

Cross Product
List Item
Range
Evaluate Surface
Shift list
Cull Nth

Introduction

The following exercise has been inspired by the covering of the roof of the Smithsonian institute by Foster and Partner. The cladding of the roof with planar quadrilateral faces shows how the tiles, in order to be planar, present a "gap" with the adjacent ones, as a consequence of the fact that non planar quadrilateral faces are covered with planar faces. The determination of the gap can be predicted in the design stage with the method proposed in this document through the application of vector operations.

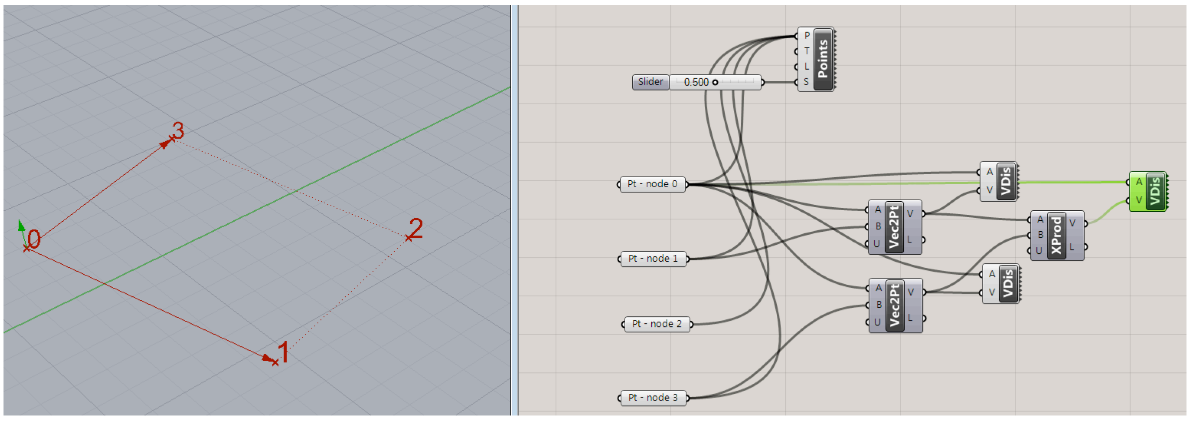
Smithsonian Institute, arch. Foster and Partners – Image courtesy of "The New Mathematics of Architecture"

Procedure

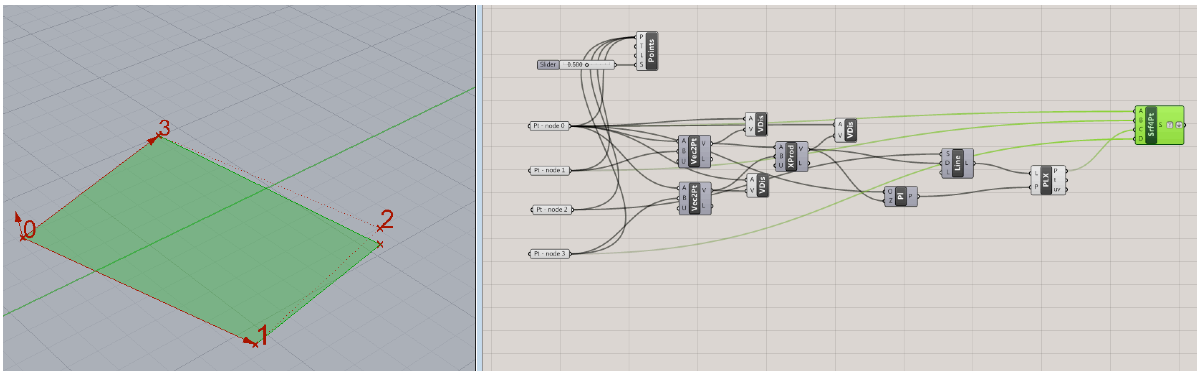
Starting document: file MP1.gh.

In the starting document each of the four "points" components (params>geometry) stores one point (node 0, node 1, node 2, node 3) that you can see previewed in the rhino canvas.

We can create a plane that passes through at least three of these four points, for example we choose node 0, node 1, node 2. To define such plane we have multiple options: our choice is to use a "plane normal" component that requires a vector perpendicular to the plane containing the three points. We create two vectors in the plane, vector **v1** (from node 0 and node 1) and vector **v2** (from node 0 to node 3) with "Vector2p" component (vector>vector). A perpendicular to vectors **v1** and **v2** will also be perpendicular to the plane. The perpendicular to two vectors can be calculated easily with the cross product operation. Execute the cross product. You can display the vectors with the "vector display" component (vector>vector), using in the A input the anchor point node 0.



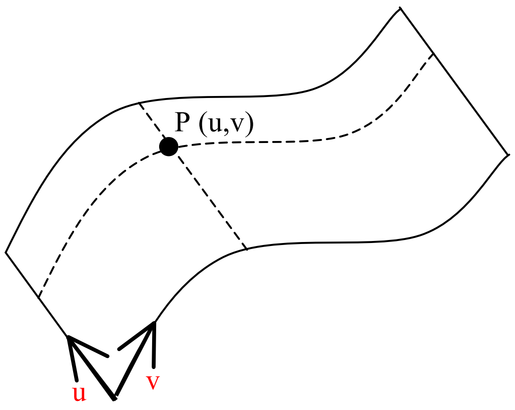
You can now create the plane passing through node 0, 1, 2 with a "plane normal" component, having origin in node 0 (but also node 1 or 3 can be seta as origin). The plane in general will not pass through node 2 (although it might happen). To make a quadrangular planar face we need to find an additional node contained in the plane, the closest as possible to node 2. The geometric construction to find this additional node is to create a line perpendicular to the plane passing through node 2 (use LineSDL component, the direction can be taken from the cross product again), and to calculate the intersection between the line and the plane (Intersect>Mathematical>Line Plane). The resulting point is used together with nodes 0, 1, 3 as inputs in the component 4PointSurface (Surface>Freeform) to create the planar quadrangular face.



Application to freeform surfaces

Preview the starting surface component (Right click>Preview) you can find in the starting file.
To apply the method to cover any freeform surface, we need to first create a square grid of points on the surface. These point will be the starting vertices to create the planar quadrilateral faces.
To create the grid of points on the surface we will use the component "Evaluate surface" (Surface>Analysis): plug the surface in the input "S".
"Evaluate surface" uses a system of local coordinates u v, so that each point on the surface can be

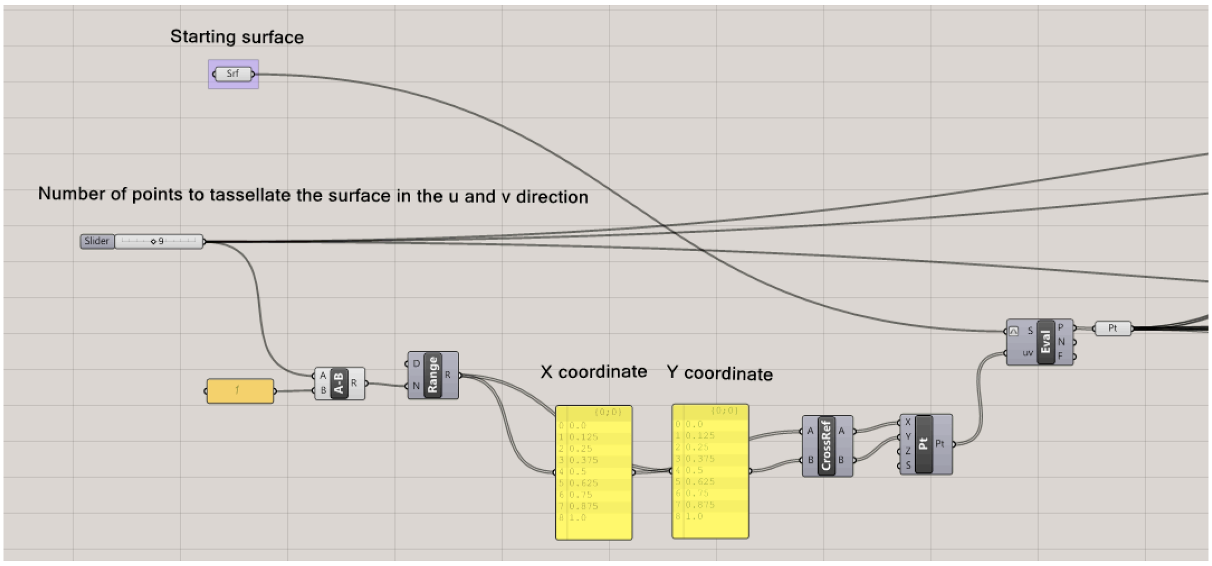
defined as P (u,v).



The uv coordinates are local coordinates that can be used to generate points on any three dimensional surface through the component "evaluate surface"

Right click on the input S and click "reparametrize", to set the total range of the u and v coordinates between 0 and 1. For the uv input we can simply use a "point" component, whose x and y coordinates are comprised between 0 and 1 (the z value is ignored) if reparametrize is selected**.**

To create a grid of points we follow a similar procedure with respect to the exercise of lecture one (see pages 2-3 of Lecture GH1 classroom exercise), however instead of the component "**Series**" to generate the coordinate values, we use a component "**Range**" (sets>sequence). "Range" is more convenient if we want to create a series of number comprised in a given range, as in this case where the points coordinate values should be distributed from 0 to 1 to tassellate completely the reparametrized surface.

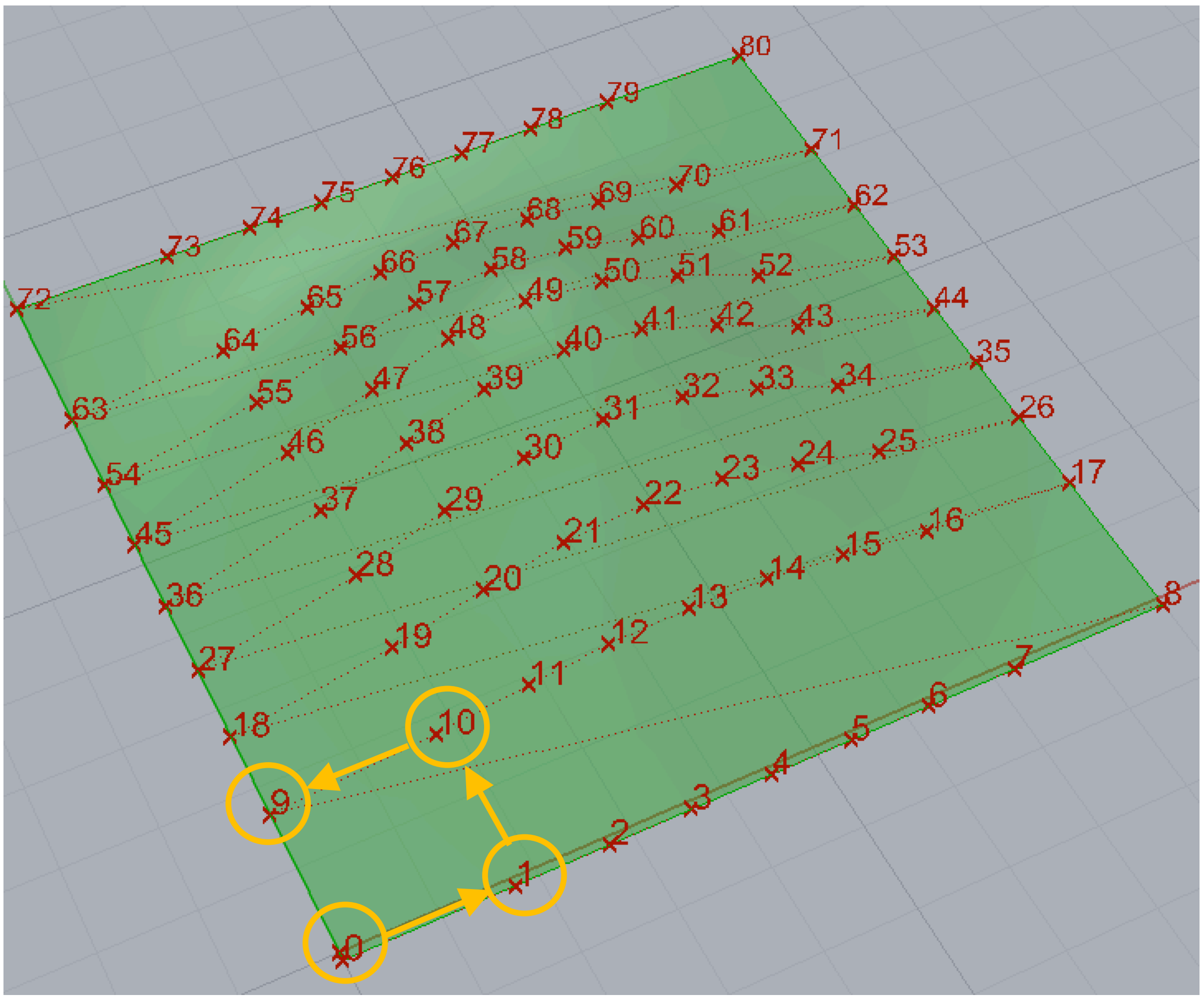


```
{0;0}
0 0.0
1 0.125
2 0.25
3 0.375
4 0.5
5 0.625
6 0.75
7 0.875
8 1.0
```

```
{0;0}
0 0.0
1 0.125
2 0.25
3 0.375
4 0.5
5 0.625
6 0.75
7 0.875
8 1.0
```

NOTE: If we plug in the N input of the "Range" component a value n, it will generate n+1 points.

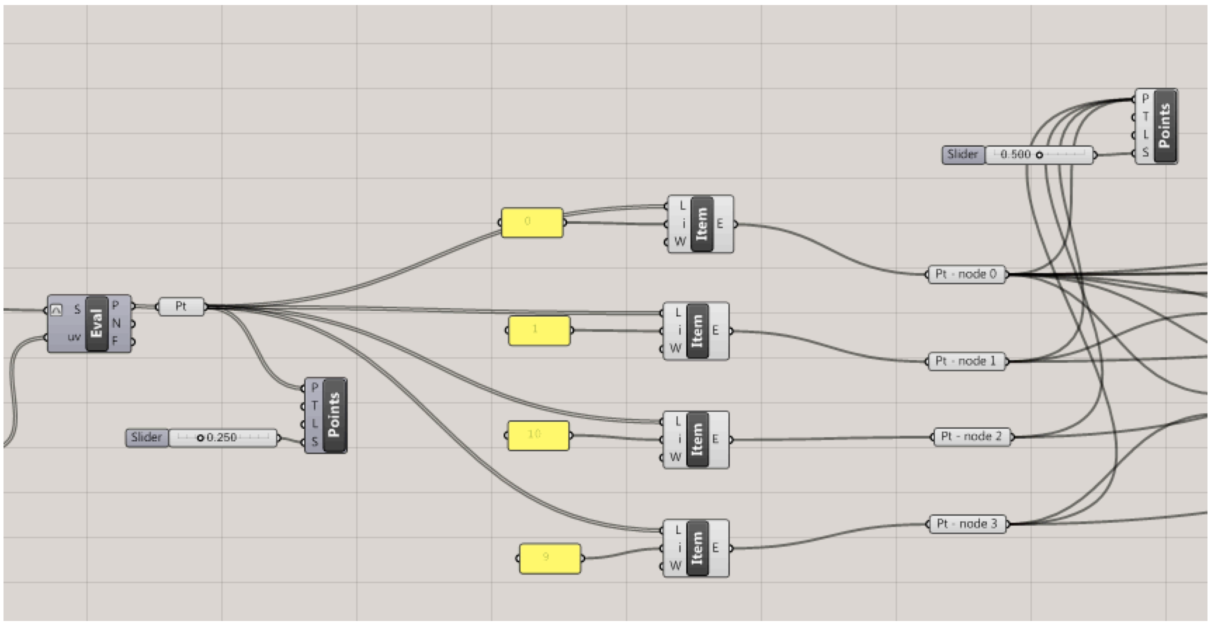
Therefore, after we create a slider of the type integer that indicates the number of desired points to tassellate the surface in the u and v direction (this value will also be used later in the definition) we have to subtract 1 before we plug it into the range component.



The surface will be populated by a grid of points. To visualize the point indexes use a "Point List" component. To start creating planar quadrangular planar faces we need to "take" from the entire list of points the four vertices of a quadrangular, and then execute the operations described at the beginning of the document.

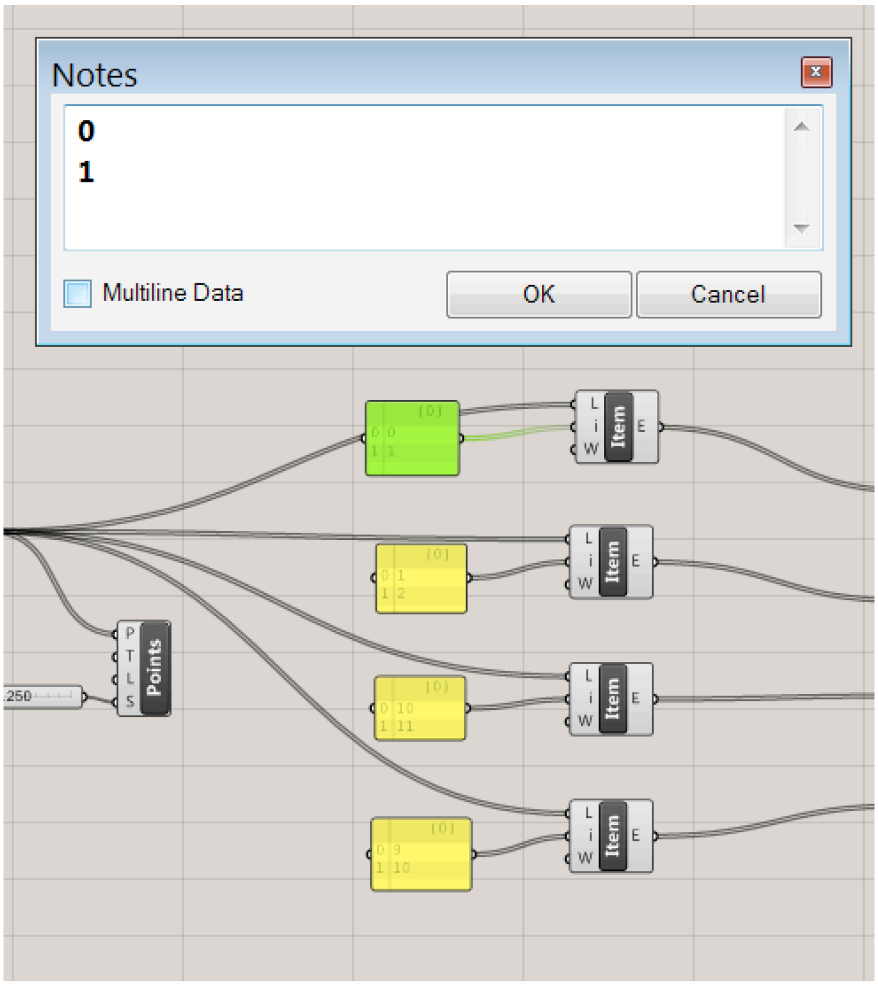
A convenient method to select a given point from a list of points is to use the component "list item" (sets>list), plugging the list of points (the output P of Eval component) in the L input and one index value (integer) in the i input.

To create the first planar face we should use 4 "list item" components, and use respectively the index 0, 1, 10, 9. Let´s create then four list item component and four panels components where you manually write the correspondent index, and connect the E output to the four point components we started the exercise with at page 2. A quadrilateral face should appear between the chosen vertices.



To create a second planar face we should add the indexes correspondent to the vertices of the second face: respectively 1,2,11,10 for the four "list item" components. Double click on the panels and add a second line with the new index, after unchecking "multiline data" in the text field panel.

We could proceed in this way for all the quadrilaterals.. However we will end up with a solution that is time-consuming and correct only for this specific case. The indexes of the points would not be correct anymore if we change the number of points on the surf



Therefore we wish to find a method which is more time-effective and that gives a correct solution whatever is the number of points/faces on the surface.

Before proceeding, we delete the four panels we used to write the indexes.

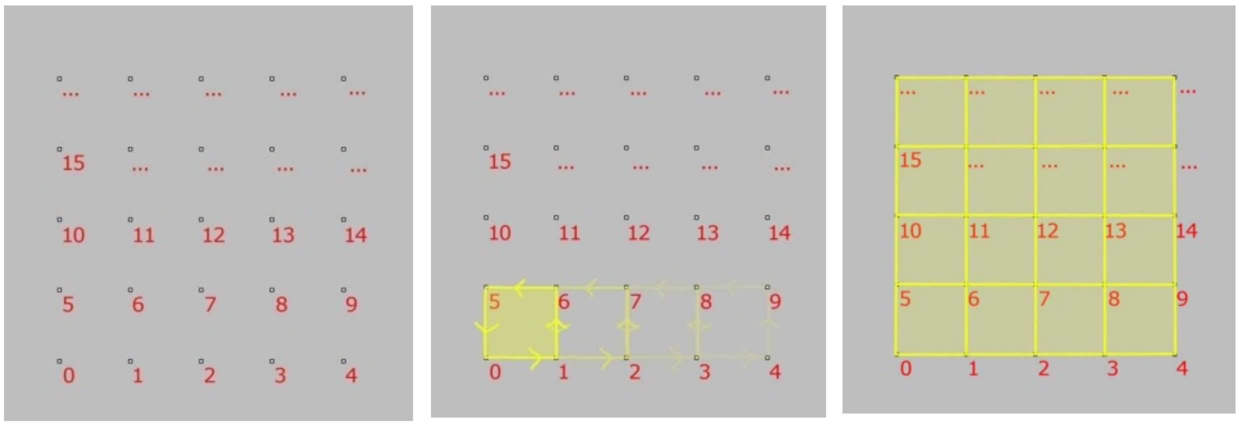


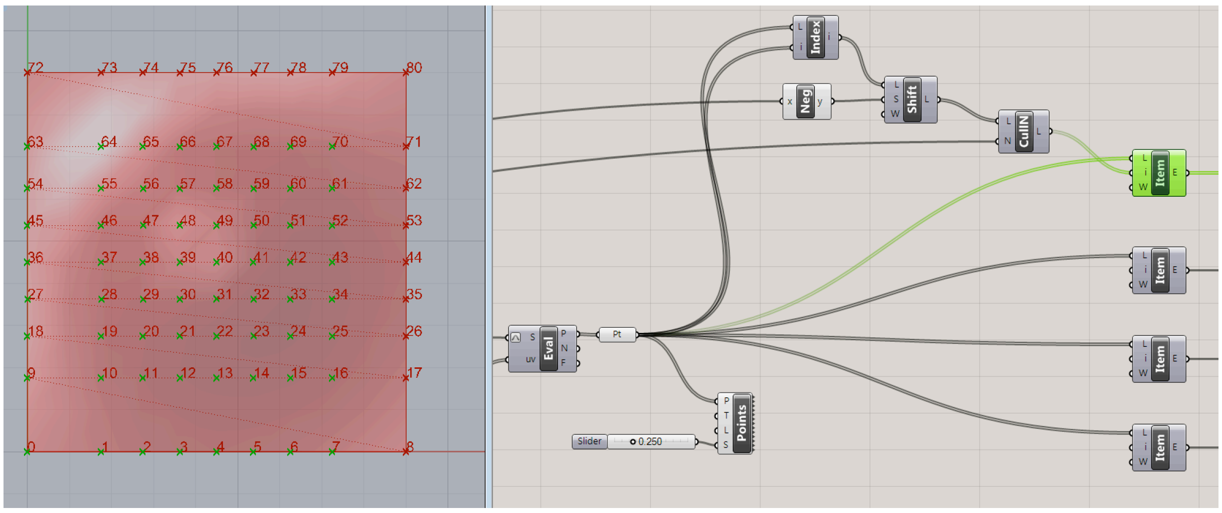
Image courtesy of Zubin Khabazi

We should be able to generate automatically the list of indexes of the vertex of each quadrilateral face to plug in the "list item" components.

We start from the definition of the indexes of the first vertices of each quadrilateral face. To do so we first set up the rule by which we select the four vertices of the quadrilaterals: we begin from the bottom left corner point to set the first vertex, then we move one point to the right for the second vertex, one point to the top for the third vertex, and one to the left for the fourth vertex.

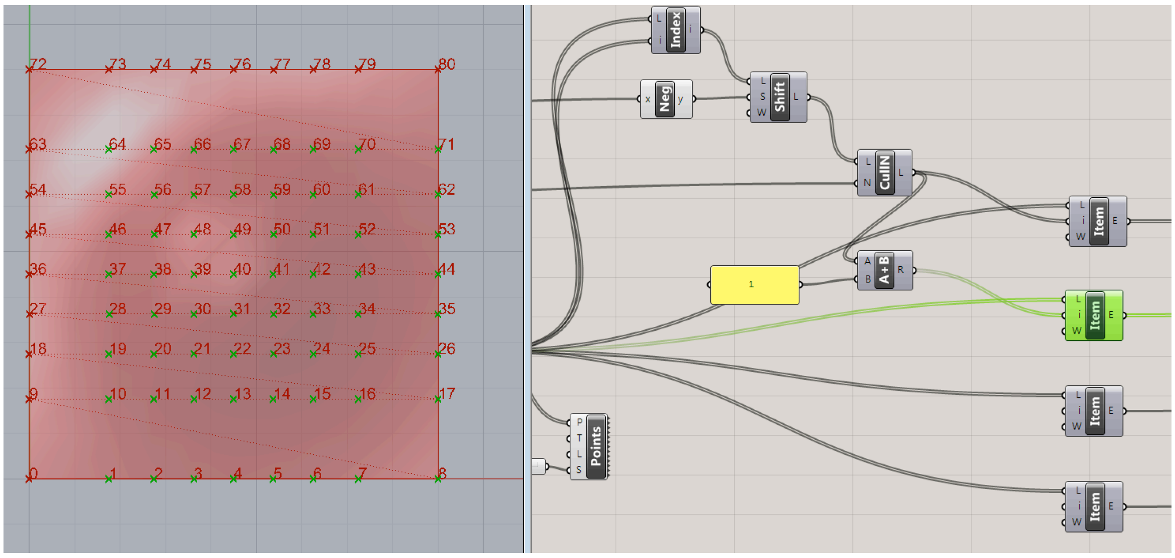
If we follow the same rationale to find the vertices of all the quadrilateral in the surface we find out that the points in the last column and in the last row cannot be the first vertex of a planar face. Therefore to select all the 'first vertices' we should first create a list of the indexes of all the points and then remove the indexes relative to the points in last column and row.

The complete list of indexes can be created with the component "Item index", and connecting to both inputs "L" and "i" the point list. The output is the complete list of indexes. To eliminate the indexes of the last row we use the component "shift list" (sets>list), plugging into L the list of indexes, and in S the number of points in the u and v direction (from the integer slider, page 4) with negative sign (math>operators>negative). This has the effect of "pushing out" of the list the indexes of the last row.



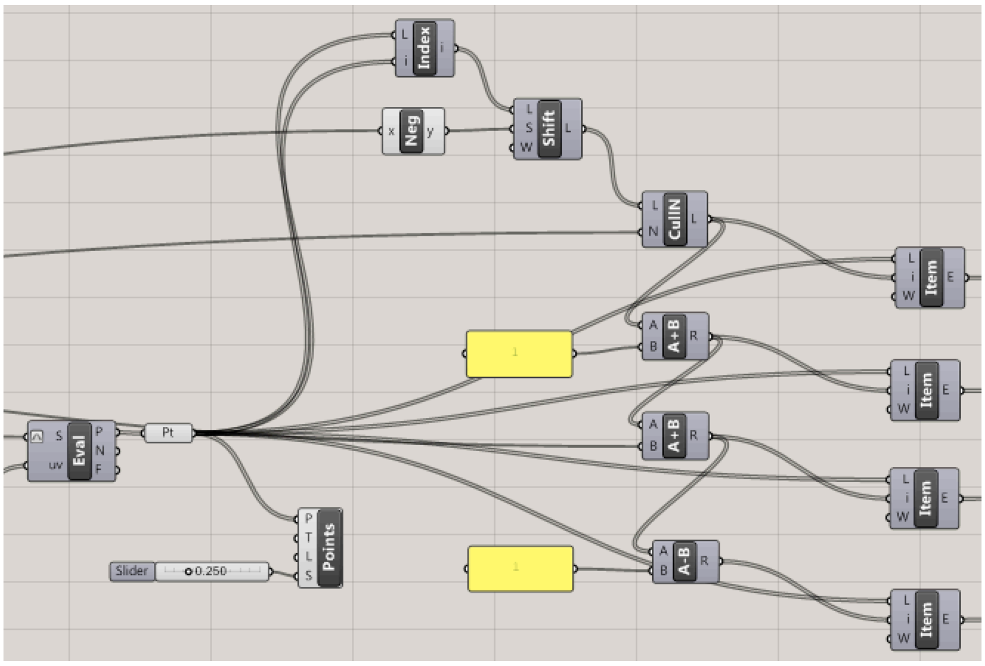
To eliminate the last row we use a “cull Nth” component (sets>sequence), that eliminate (cull) every Nth point in the list, where N is the number of points in the u and v direction (from the integer slider, page 4).

Now we proceed with the indexes of the second vertex of each quadrilateral face. Each ‘second vertex’ is immediately to the right of the ´first vertex´, therefore its index will be equal to the index of the first point +1. Therefore we simply use an addition component with the list created previously and we plug the new list in the second “list item” component



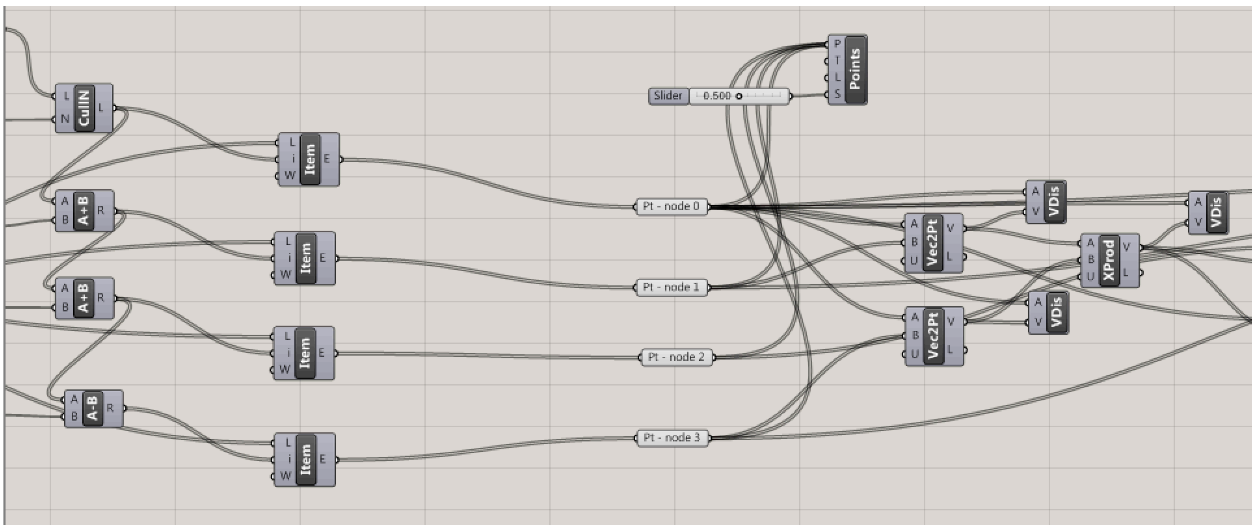
The third vertex is one row up, therefore its index is obtained by adding to the indexes of the second vertex the number of points per row (you have this value defined parametrically with the integer slider, page 4). This has the effect of selecting to the point in the upper row.

The fourth vertex is one point left of the third vertex, and its index is therefore obtained by subtracting one to the indexes of the third points



‘

By plugging these four List item components to the four point components we started the exercise with at page 2, planar quadrilateral faces should appear covering the whole surface. You can edit the number of quadrilateral faces by changing the number of subdivision of the range component.



You can use this definition for any surface. You create the surface, and then use it instead of the provided geometry. To convert the grasshopper preview into actual rhino geometry, right click on the component containing the desired grasshopper component and click bake.