Det Teknisk-Naturvidenskabelige Fakultet Første Studieår
AALBORG UNIVERSITET
Arkitektur Og Design

MATEMATIK OG FORM

18 February 2013 - Lecture GH1 (in English)
Vector operations in Grasshopper

Group 1	12:30-13:30	Lecture at Auditorium 7 *Lecturer*: Dario Parigi
	13:30-15:15	Task Check at group room *Teachers*: Martin Raussen, Jesper Christensen, Mathias Jensen
Group 2	13:30-14:30	Lecture at Auditorium 7 *Lecturer*: Dario Parigi
	14:30-16:15	Task Check at group room *Teachers*: Dario Parigi, Jesper Christensen, Mathias Jensen

Before the lecture
-All students should come to the lecture with your laptop with Rhinoceros and Grasshopper installed.
Rhino 5.0 Evaluation Download: http://www.rhino3d.com/download/rhino/5.0/evaluation
Grasshopper http://www.grasshopper3d.com/page/download-1
- Al students should read the "Grasshopper Learning Material" document

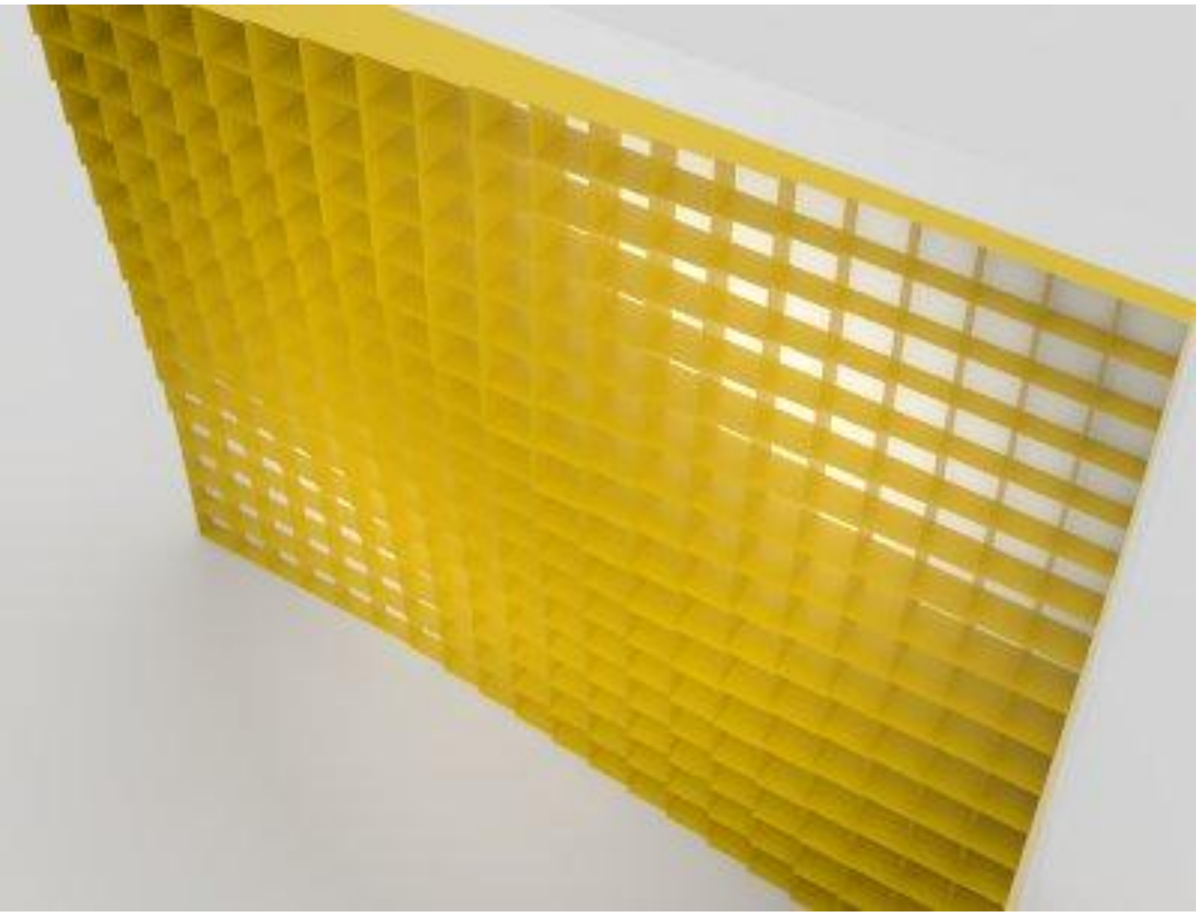
Aims and contents:
The lecture goal is to perform the basic vector operations with Grasshopper, and to create an architectural component using vector data taking advantage of the potential of the parametric modelling environment.

Lecture schedule
-Getting started and introduction to Grasshopper
-Vector operations
-Architectural applications

Tasks
group room task: see document
home assignment: taking inspiration from the group room exercise, create an architectural facade in Grasshopper using vectors components and/or vector operations.
You can use some time during the 'workshop slot' on Tuesday morning the 1st of May to develop your proposal

Literature
R. Issa, Essential Mathematics for computational design, 2nd ed. (pages 1-12)
Woojae Sung, Grasshopper Learning Material
Grasshopper Primer_Second Edition_090323 (pages 1-22)

LECTURE GH1 CLASSROOM EXERCISE



What you will learn

1. create and manipulate large set of vectors and data
2. use large sets of data to create an architectural component/façade

List of relevant components used

Point XYZ

Cross reference

Series

Vector2Pt

Vector unit

Introduction

The following exercise has been inspired by the article published on http://www.designedbyenergy.com/site/?p=202 focused on the development of a design tool for shaping a shading device with respect to given light and energy saving goal performance. The design tool explores different configurations of variable extrusion depths on the surface and with the use of optimization strategies chooses the best performing solutions.

With the following exercise we will create a shading component based on a square grid with the possibility to vary the extrusion depth with attractors. We will not use performance analysis and optimization strategies, however we will create a framework where such analysis can be easily implemented.

Procedure

First we create a square grid of point as the basis for the construction of the shading device. We use the components “**PointXYZ**” (Vector>Point), a component “**Series”** (Sets>Sequence), two sliders, and the component **Cross Reference** (Sets>List). The component Point XYZ create points from X, Y, Z coordinates (inputs on the left of the component).

We generate the numerical values of the X and Y coordinates of the points that we want disposed in a regular quadrangular grid: because the points are equally spaced the values of their coordinates can be easily generated as a sequence of numbers with the component **Series** (that we should now insert in the grasshopper canvas).

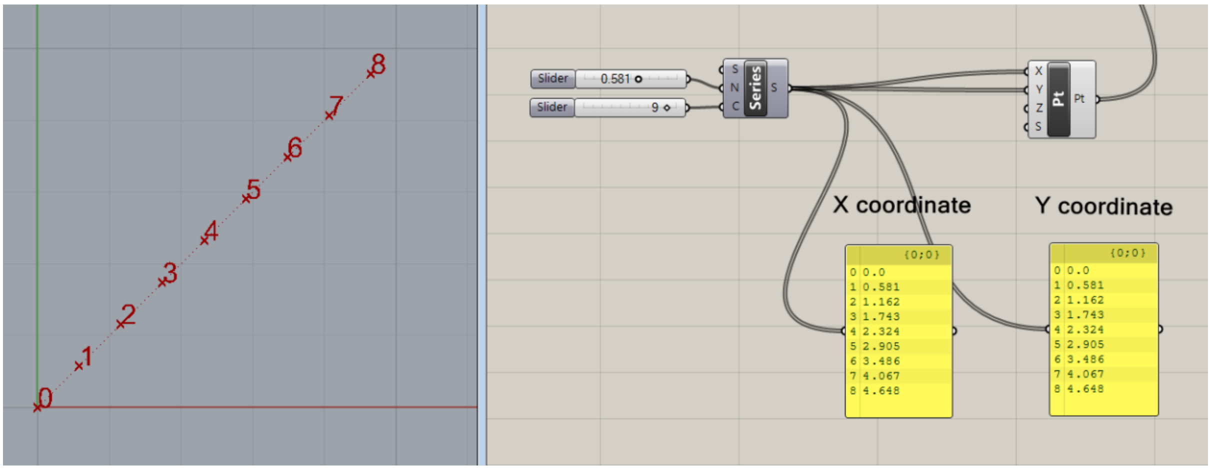
We first generate a list of number to be used for the x coordinates. In the input of the component

Series we plug-in with a **number slider** the step size of each subsequent number in the series – input N- (this will correspond to the spacing of the points in the x- direction) and the number of points in the series – input C- (this correspond to how many points we want in the horizontal side of the grid).

We use sliders for both input, however for the C input the slider should be of the type integer (right click to select that option), and the max value increased to, for example, 10 (right click on the slider>values>max).

We should generate a similar list for the values of the y coordinates. Because we want a square grid, the spacing and the number of points in the y direction will be the same as in the x direction, therefore we can use the same list created for the x coordinate also for the y coordinate.

Now we can use the list of numbers as it is in the X and Y input of the “pointXYZ” component.



```
{0;0}
0 0.0
1 0.581
2 1.162
3 1.743
4 2.324
5 2.905
6 3.486
7 4.067
8 4.648
```

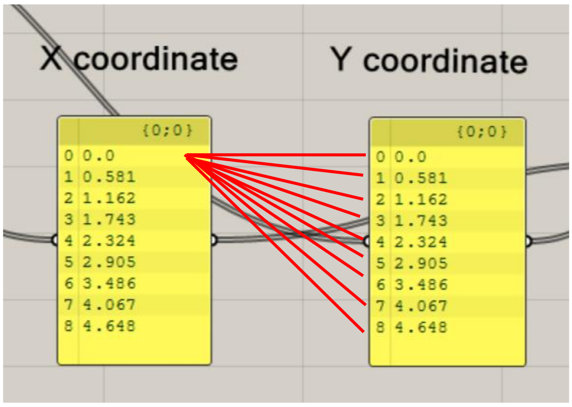
```
{0;0}
0 0.0
1 0.581
2 1.162
3 1.743
4 2.324
5 2.905
6 3.486
7 4.067
8 4.648
```

However we will obtain the generation of only 9 points on the diagonal of a square grid, because the component Point XYZ couples **one to one** the values received in the X and Y input: a first point is created with the first X value coupled with the first Y value, a second point is created from the second X value coupled with the second Y value, etc…The z value is by default 0

Use a **Point list** component (Vector>Points) to visualize the point indexes, using a slider to determine the text size in the input S, and the output of Point XYZ in the input P

To obtain a grid instead we need to couple each of the values of the X coordinate with each of the values of the Y coordinate.

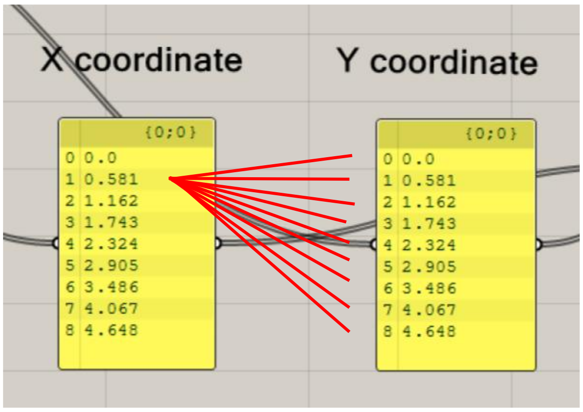
For X coordinate 1:



```
{0;0}
0 0.0
1 0.581
2 1.162
3 1.743
4 2.324
5 2.905
6 3.486
7 4.067
8 4.648
```

```
{0;0}
0 0.0
1 0.581
2 1.162
3 1.743
4 2.324
5 2.905
6 3.486
7 4.067
8 4.648
```

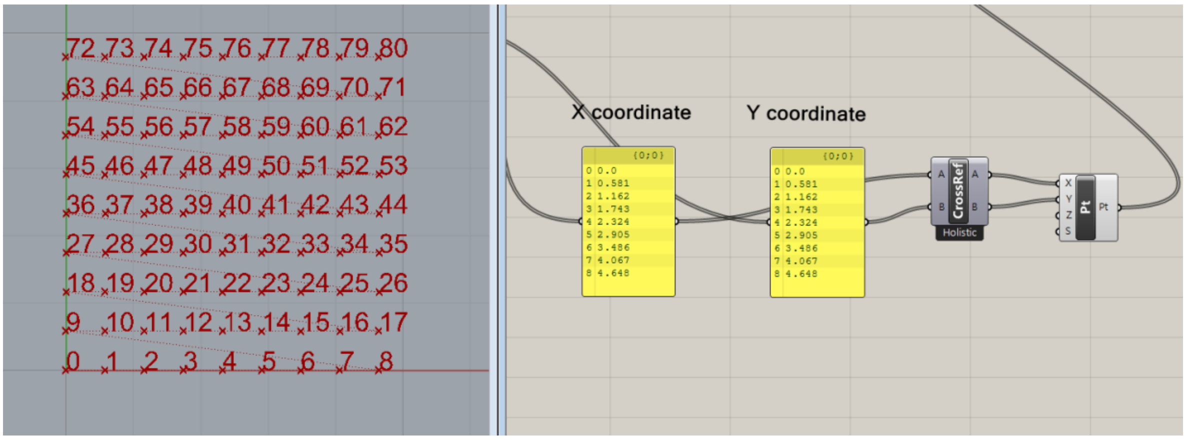
For X coordinate 2, and so on for all x coordinates:



```
{0;0}
0 0.0
1 0.581
2 1.162
3 1.743
4 2.324
5 2.905
6 3.486
7 4.067
8 4.648
```

```
{0;0}
0 0.0
1 0.581
2 1.162
3 1.743
4 2.324
5 2.905
6 3.486
7 4.067
8 4.648
```

The component Cross reference (Sets>List) conveniently creates in a single steps all the possible combinations between couples of x y coordinates.



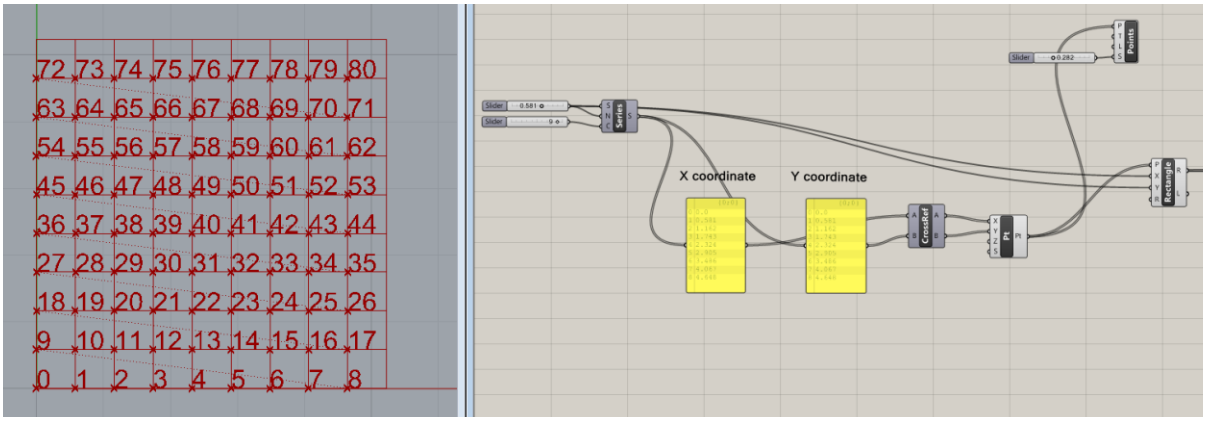
```
{0;0}
0 0.0
1 0.581
2 1.162
3 1.743
4 2.324
5 2.905
6 3.486
7 4.067
8 4.648
```

```
{0;0}
0 0.0
1 0.581
2 1.162
3 1.743
4 2.324
5 2.905
6 3.486
7 4.067
8 4.648
```

A grid of points should appear in the Rhino canvas. You can right click on the component Cross Reference and experiment with the different cross reference options (Diagonal, Coincident,etc…), but then return to the default option Holistic.

Create a rectangle starting from each point with the component Rectangle.

Use the slider previously defined for the spacing (N input) of the Series component to determine the X and Y dimension of the rectangle, and the points for the P input.



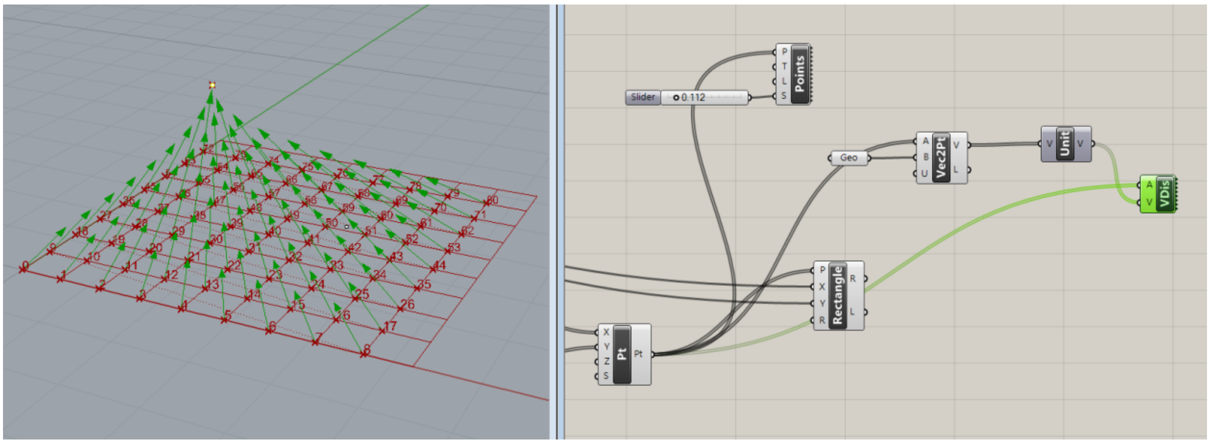
Create the architectural component from the square grid

Go to the Rhino interface and draw an external point, that we call P1, in the Rhino canvas with the “Single point” command in Rhino (not in Grasshopper!), then move back to the grasshopper interface and reference the point just created inside Grasshopper by using a “**geometry**” component (**params>geometry**), and right click>set one point (the rhino canvas will come into foreground automatically), and select the point just created.

Create a vector from the square grid points and the external point with the component (Vector>Vector>Vector2Pt).

Unitize the vectors (Vector>Vector>Unit Vector) to set the length of the vectors to 1.

To visualize the vectors plug the vectors into the V input of the component vector display (Vector>Vector>Vector Display), and the base points in the anchor point input (A input).



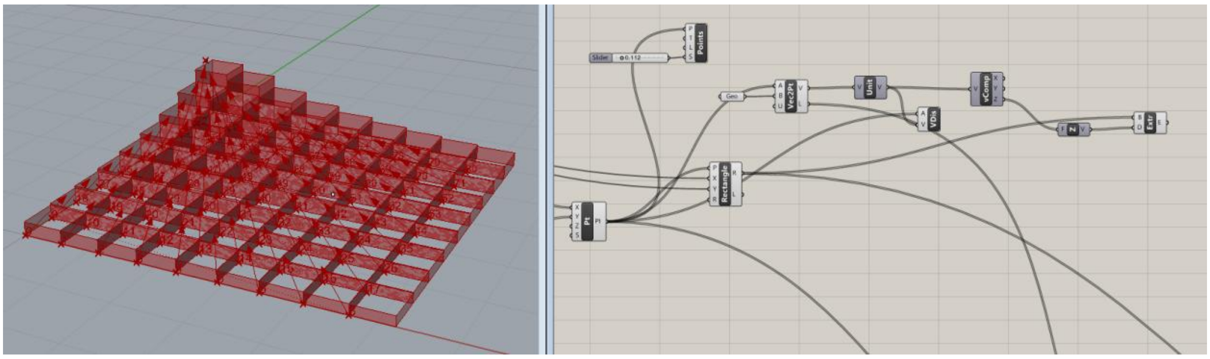
NOTE: To increase or decrease the number of squares in the grid just modify the C input of the series component defined at the beginning of the exercise.

Now we create a solid geometry starting from the rectangles using “extrude”, and the height of the extrusion will be determined by data extracted from the vectors just defined.

The extrude component (**surface>freeform>extrude**) accept as inputs the geometry, in our case the rectangles defined with the “rectangle” component, and the extrusion direction and height, defined with a vector, in the D input.

For the D input we first decompose the vectors in the x y z components,

Then we create a “unit z” vector (vector>vector) because the z is the direction we want to create the extrusion and plug it into the D extrude input. To create variation in the extrusion height we can plug in the unit z vector, for example, the z component of the vectors.



Try to experiment by using as extrusion height, for example, the sum of the x and y vectors components. In this exercise we are using abstract values for the extrusion. However in a design context, the values you use can be related to specific design requirements. If you imagine the component being an architectural façade, you can for example relate the extrusion height to variable amount of passing light, or visibility in different parts of the façade.

Thresholds and further data manipulation

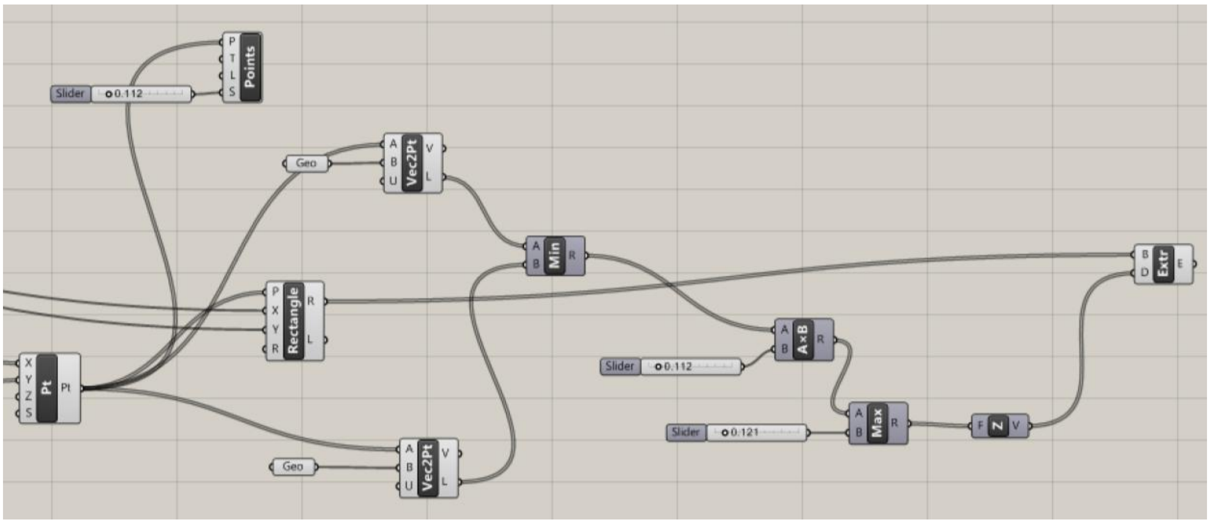
With this last part of the exercise we experiment with other kinds of data manipulation before it is used as value for the extrusion vector. You can delete or disable the previous part of the exercise from the “unit vector” component and the following components.

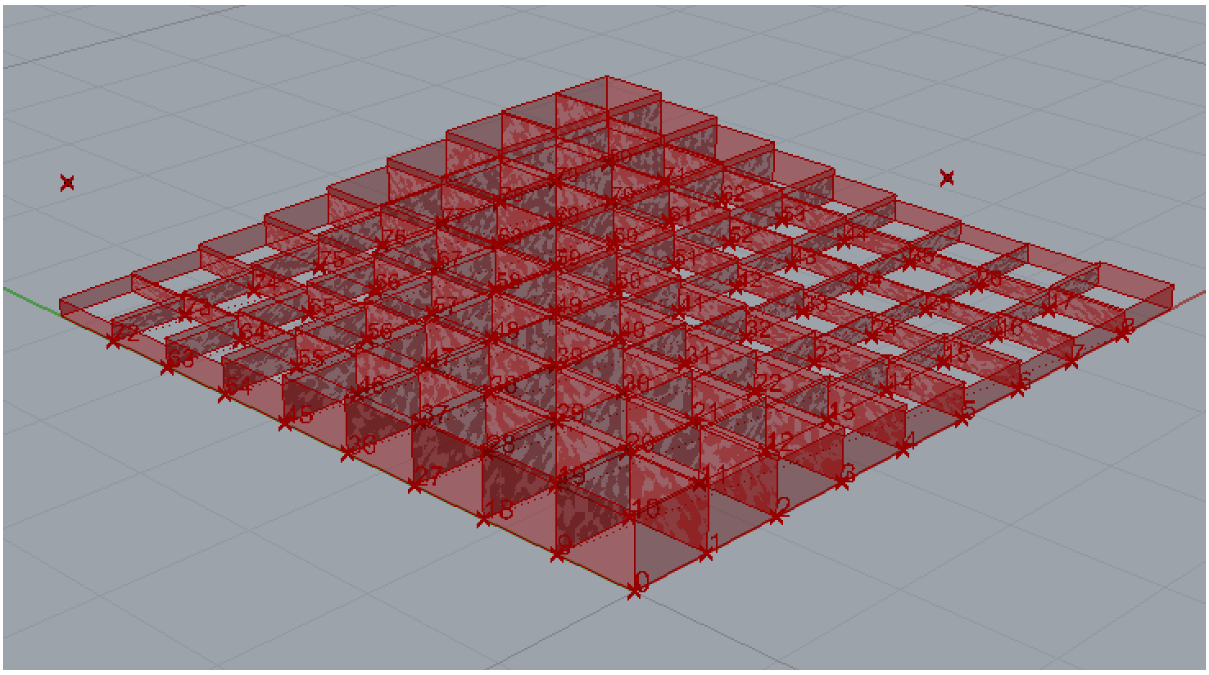
We create another external point, P2, in the rhino canvas, and we reference it inside grasshopper with a geometry component. We create vectors from the point grid to the point P2. The L output of the “Vec2Pt” component list the length of each of the created vectors. Now we take the L output from these vectors and the L output from the vectors created with the previous point P1 and we plug them into a “min” component (math>util). For each of the points on the grid there is a vector towards P1 and another towards P1. The “min” component will choose for each point the length of the shorter between the two.

Multiply the “min” output with a scaling factor to reduce its values: use a slider with values comprised form 0 to 1 (the default values)

Now we set up a threshold, a value below which these value should not go. The threshold value can be defined with a slider (you decide the range values..), then plug the threshold and the scaled values to a "max" component (math>util). The component will choose the threshold value if the scaled values are minor than that.

Now use the resulting values as input in the unit z vector you use for the extrusion: you can experiment with different combinations of the values of the scaling and the threshold value.





These experiments are still of an “abstract” nature, but you can think at the potential to use, in a design situation, data coming from specific design requirements or analysis (specific site conditions, light analysis, structural analysis, etc..), that can be filtered/manipulated before you can use them as values to determine the final geometry of your architectural component/building/site planning.