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

MATEMATIK OG FORM

10 May 2012 - Lecture 11 (in English)
Matrices and Transformations in Grasshopper

Group 1	8:15-9:15	Lecture at Auditorium 3
		Lecturer: Dario Parigi
	9:15-11:00	Task Check at group room
		Teachers: Martin Raussen, Poul Henning Kirkegaard, Dario Parigi
Group 2	9:15-10:15	Lecture at Auditorium 3
		Lecturer: Dario Parigi
	10:15-12:15	Task Check at group room
		Teachers: Martin Raussen, Poul Henning Kirkegaard, Dario Parigi

Aims and contents:
The lecture goal is to perform transformations in Grasshopper (moving, rotating, projecting, scaling) using the built-in components. It will be shown how to reveal the transformation matrices behind those operations. Before the task check it will be explained how grasshopper manages data in "tree structures".

Lecture schedule
-Transformations in Grasshopper
-Trees data structure

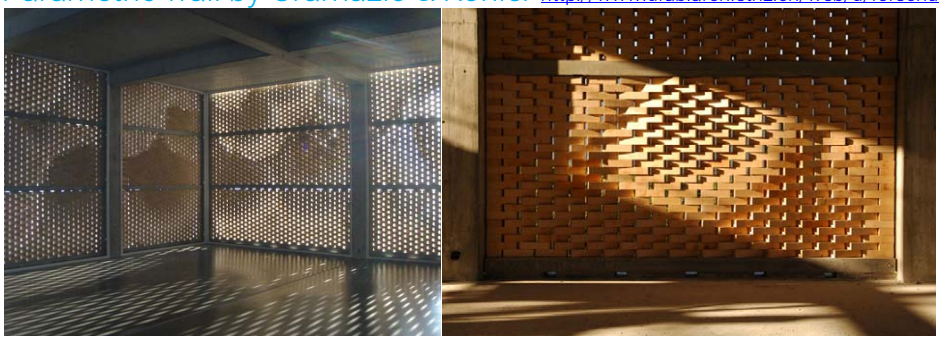
Tasks
group room task: see document

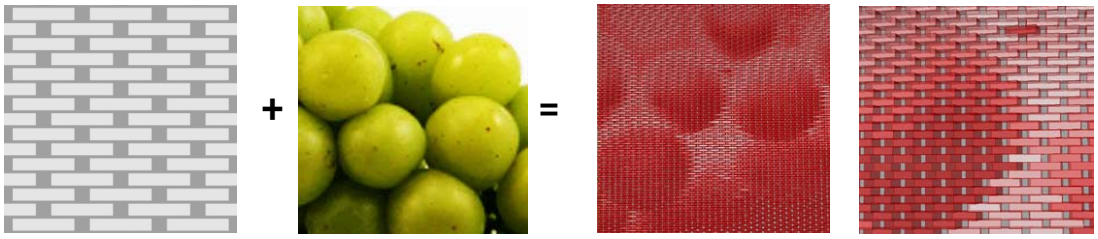
Literature
R. Issa, Essential Mathematics for computational design, 2nd ed. (pages 16-20)
Woojae Sung, Grasshopper Learning Material (pages 10-11)
Grasshopper Primer_Second Edition_090323 (pages 36-39)

MATEMATIK OG FORM

10 May 2012 - Lecture 11 Tasks

Parametric wall by Gramazio & Kohler http://www.dfab.arch.ethz.ch/web/d/forschung/52.html





what you learned in lecture 4

- basic grasshopper operations
- manage large set of elements
- apply mathematical knowledge to architectural applications

what you will learn in lecture 11

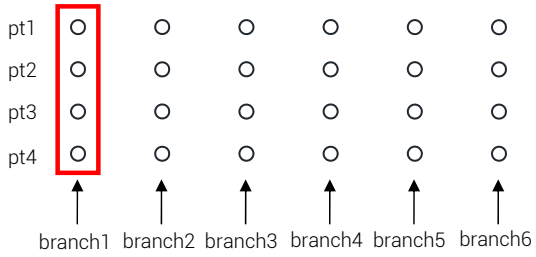
- to use and apply transformations
- to manage trees data structures

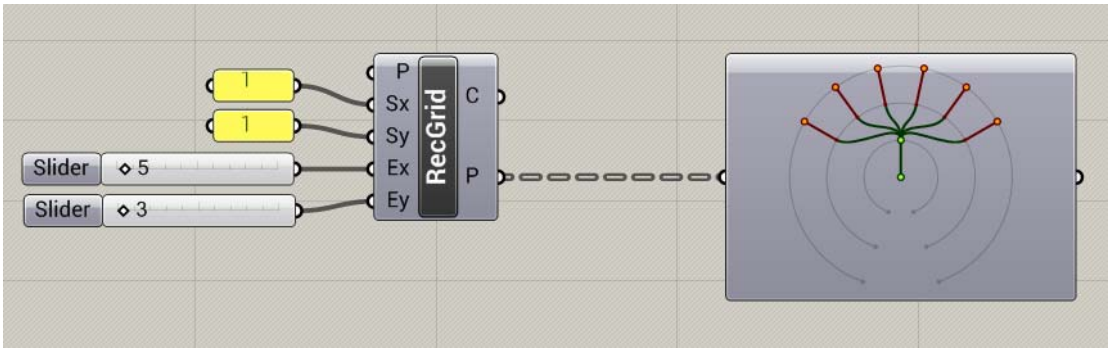
2) Additional assignment

Use and apply different types of transformations using another type of data source

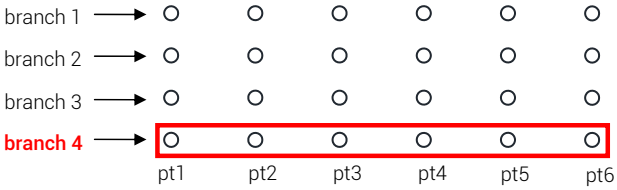
Intro: tree structures

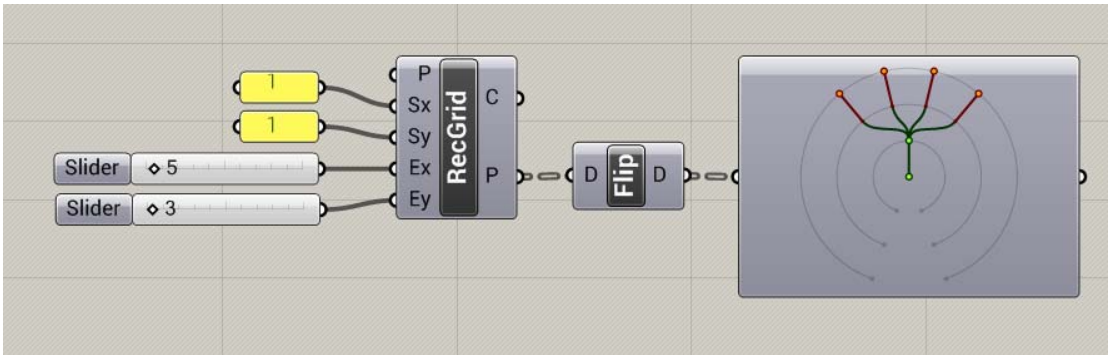
GRASSHOPPER works with trees data structures.
A set of 24 points created with RECGRID orders the points into 6 branches with 4 points each (use sliders of type INTEGER right clicking on it). With the component PARAM VIEWER (**params>special**) it is possible to list the branches of the trees, and and double clicking to visualize the tree of the data structure.





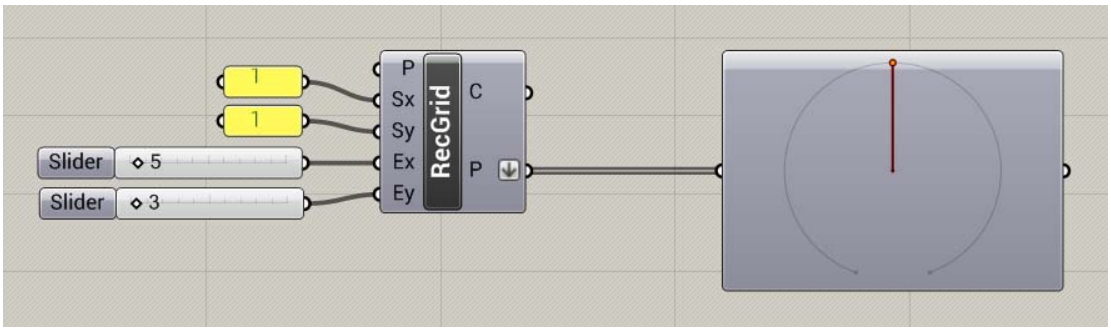
with the command FLIP MATRIX (**sets>tree**)it is possible to "flip" the tree data structure, and to order the points nstead into 4 branches with 6 points each



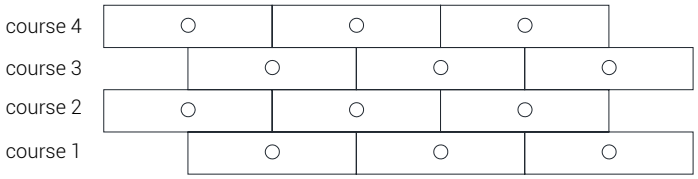


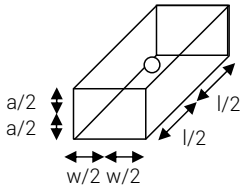
It is possible to eliminate the branches with the command **FLATTEN (right click on the P output of RECGRID)** : in this way the points are ordered in a single branch one after the other

pt1 pt5 pt9 pt13 pt17 pt21
pt2 pt6 pt10 pt14 pt18 pt22
pt3 pt7 pt11 pt15 pt19 pt23
pt4 pt8 pt12 pt16 pt20 pt24

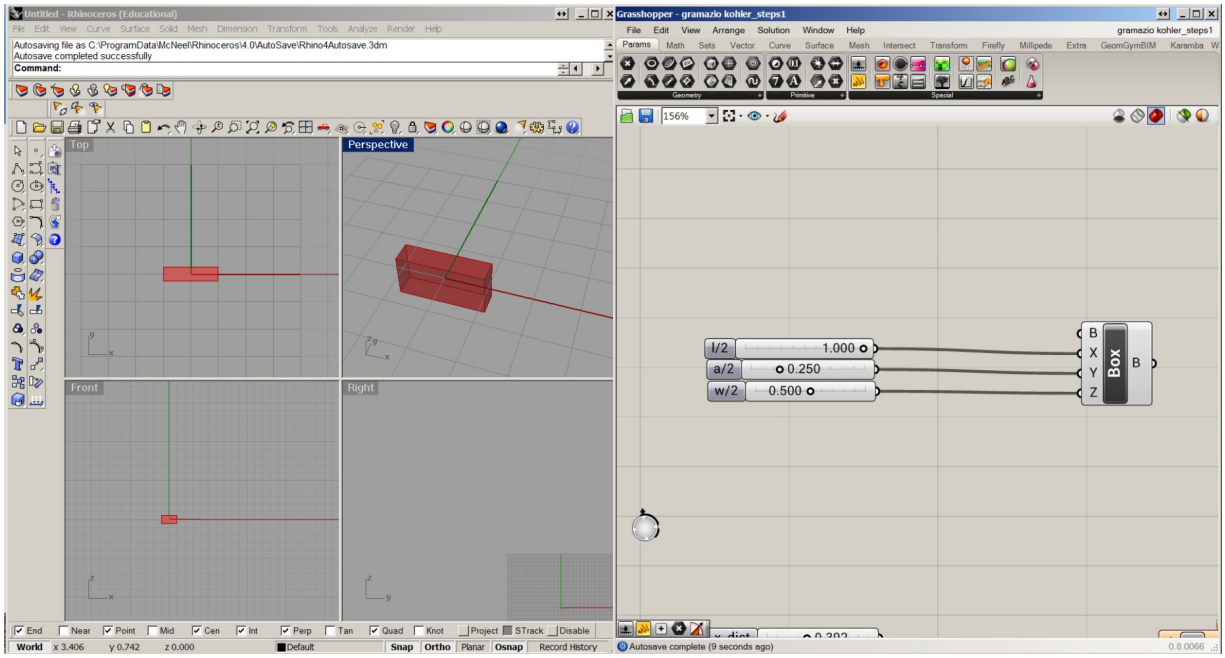


Create a brick wall

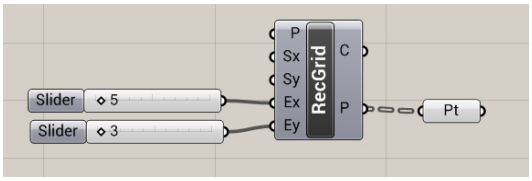


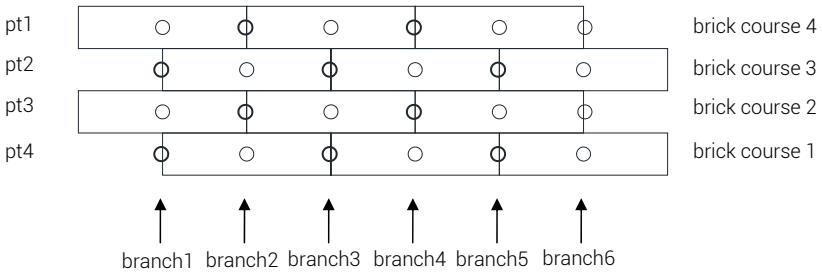


Each brick is drawn starting from its central point wit h the component CENTERBOX (**surface>primitive**), and the dimensions to input are half the total dimensions.

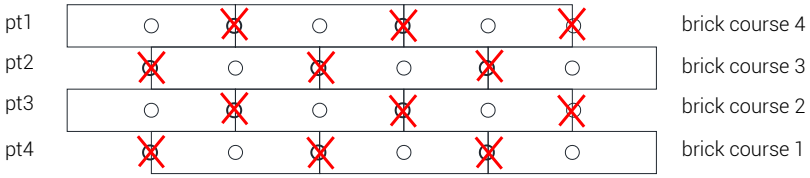


We create the central points of the bricks from the previously created RECGRID component .

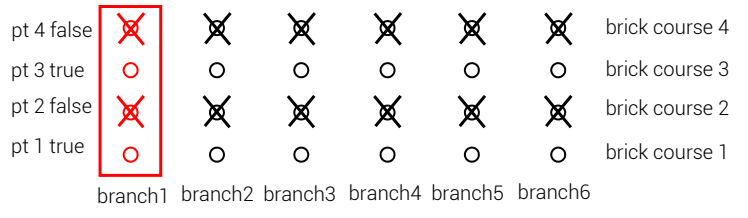


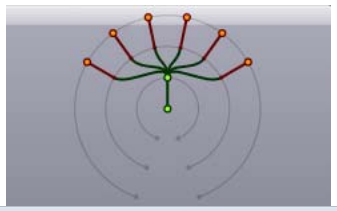


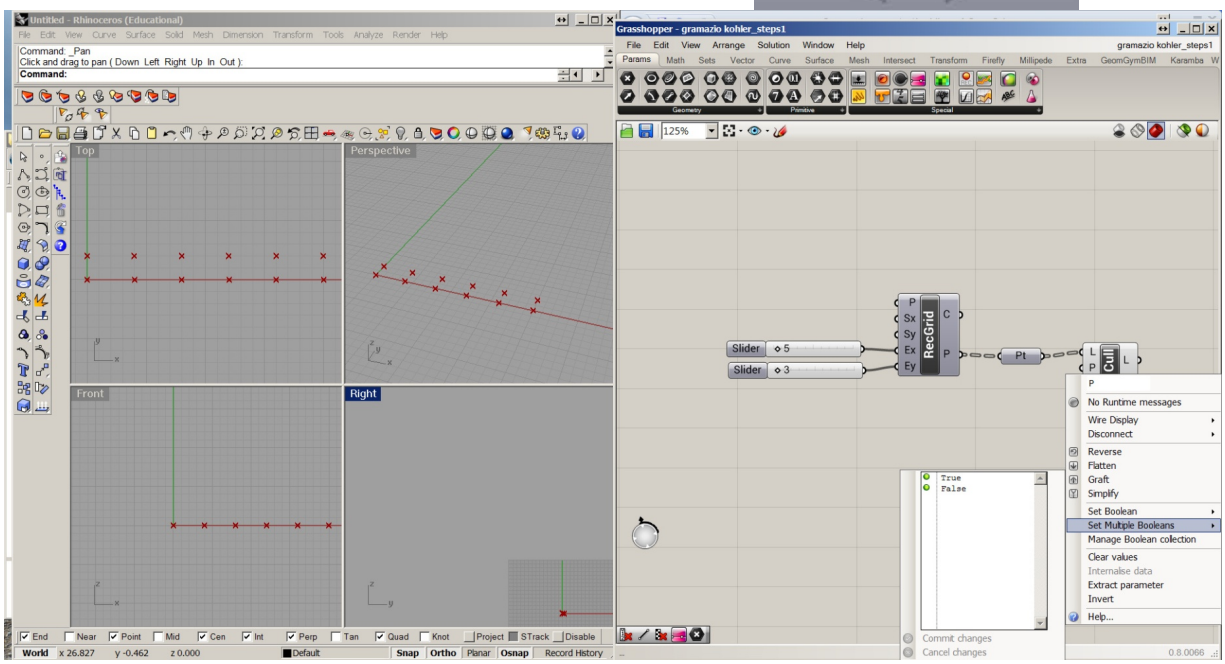
However we have to selectively eliminate some points in each row according to the following scheme in order to have one point per brick.



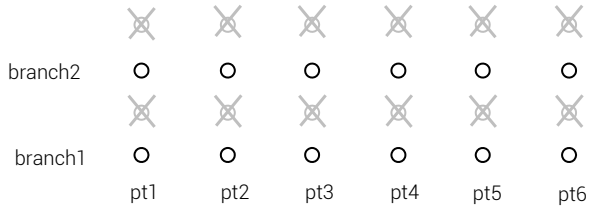
We have to operate separately on the odd/even curses. First we will operate on the odd curses, and to isolate the odd courses we use a **cull pattern (sets>sequence)** component with the pattern "**true; false**" (right click on P>set multiple booleans)**.** This will have the effect to remove the points in each branch according to the repeating pattern (true , false) where "false" delete the point and "true" keep it.

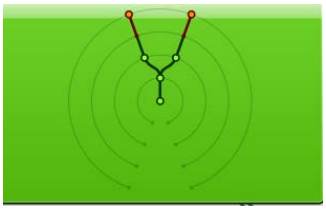


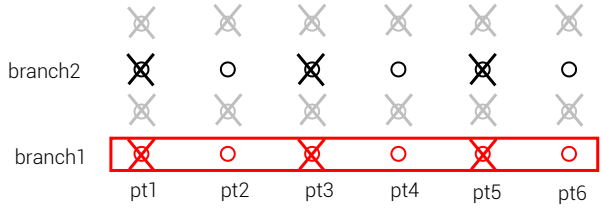


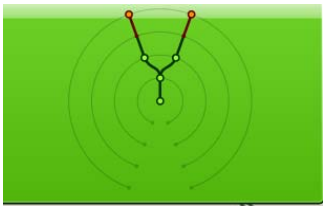


The next step is to flip the structure again and then to apply a cull pattern that eliminate the points with a pattern "**false; true"**

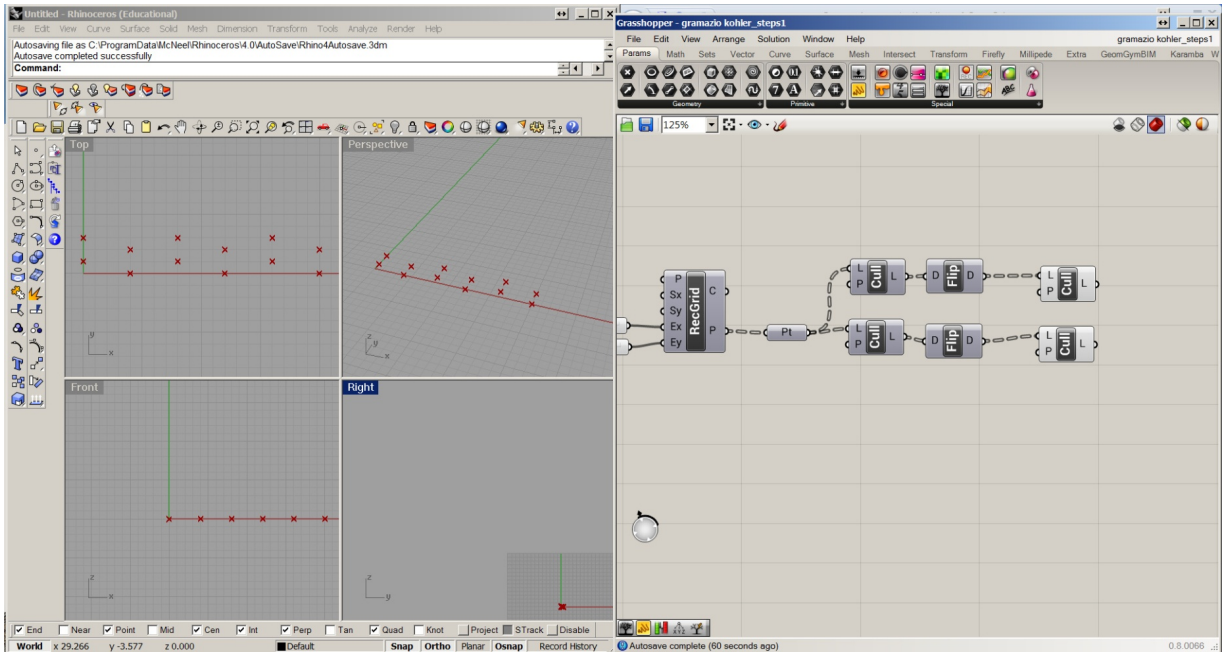




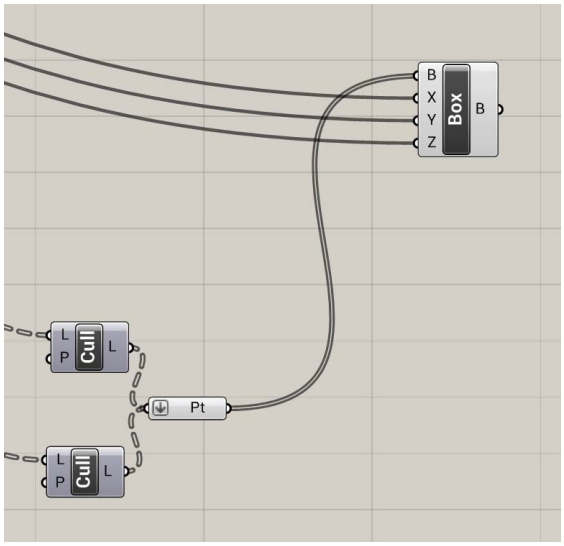




TO DO :
REPEAT THE SAME PROCEDURE WITH THE EVEN COURSES APPLYING THE APPROPRIATE CULL PATTERN

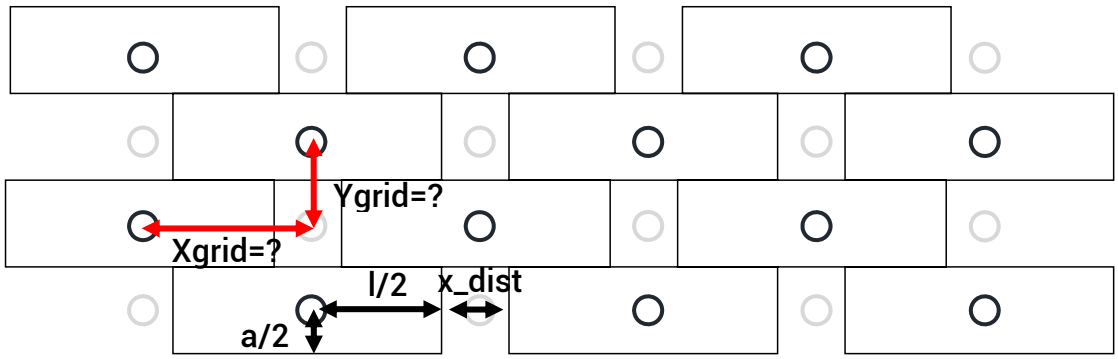


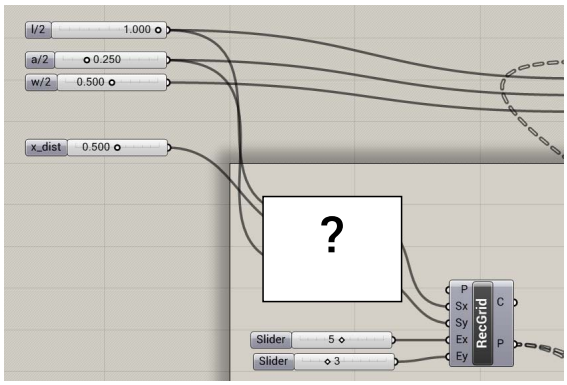
Now you can merge the two sets containing the central points of the bricks of the odd and even courses in a point component (**params>geometry**) and flatten the list (right click on the point component) . We don't need anymore the tree data structure..
Then connect the point to the box component representing the brick that we created before.



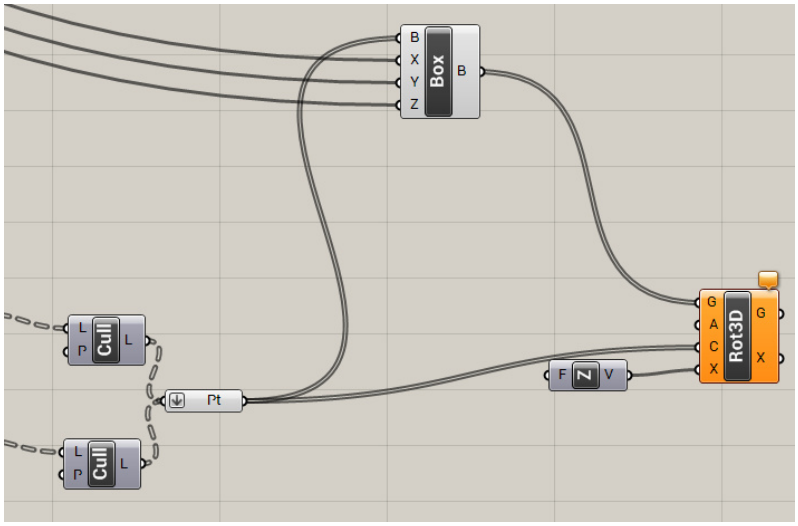
At every point is now associated a brick. However it is necessary to define the correct spacing of the initial grid to ensure the contact of the bricks in the y direction, and a spacing (x_dist) in the x direction.

TO DO :
Define the spacing in the x and y direction of the RECGRID component (Sx and Sy), replacing the values that we gave at the beginning, using the dimensions of the brick as starting values



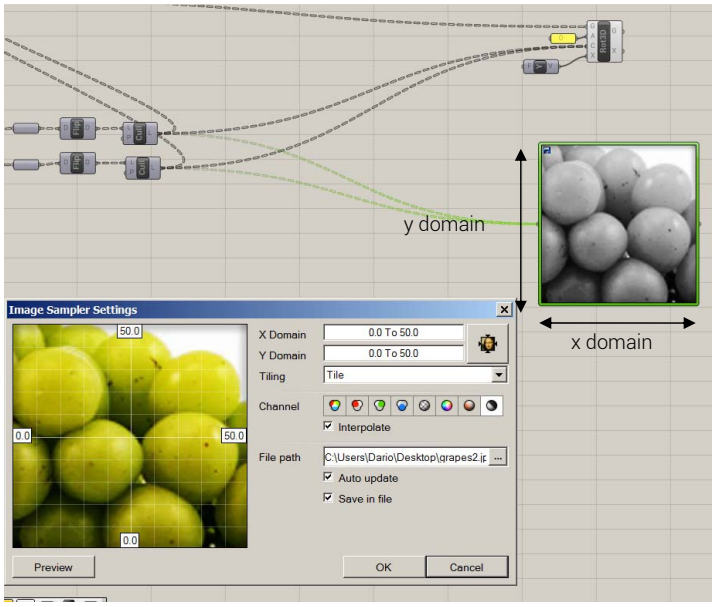


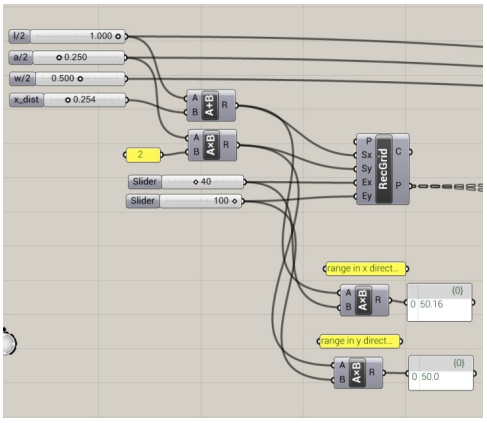
Now we are going to operate transformations on the wall .
To follow our example we start with rotations, using a **rotate 3d** component (**transform>euclidean**), setting the boxes as geometry, a unit z vector as rotation axis, and the central points as center of rotation.

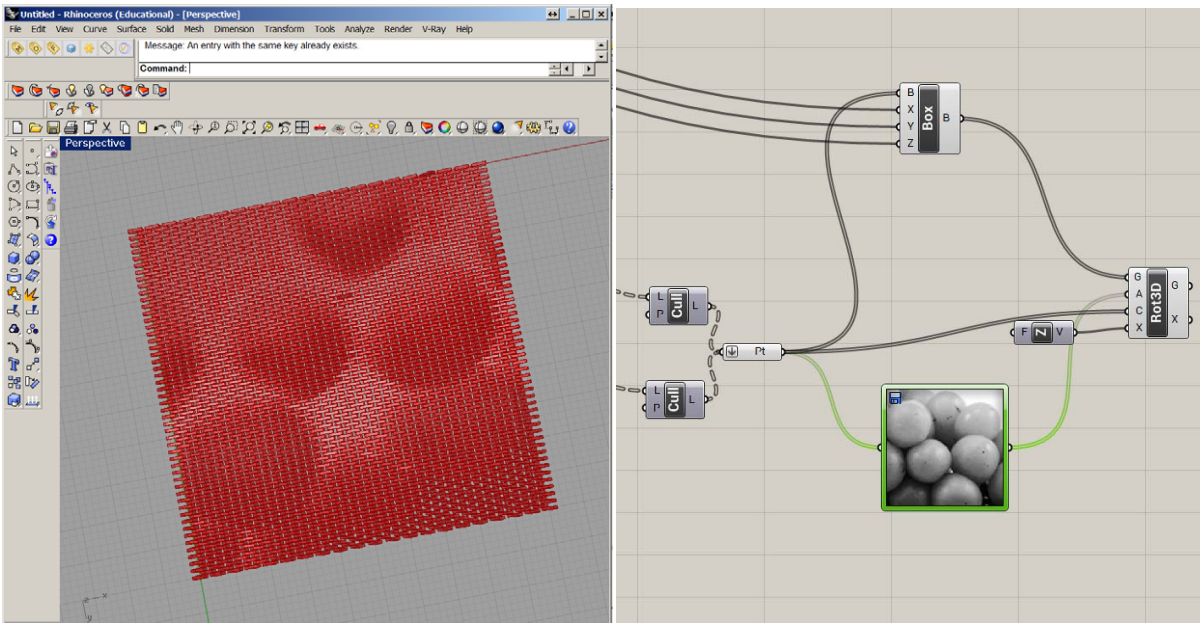


It is possible to input the rotation of each block in radians using values inherited from other sources.
In our example we use a **image sample** (**params>special**) component. It accepts points as inputs , and it evaluates various characteristics of an image (saturation, brightness, etc..) in that positions. By double clicking on the component we can set the image path and edit the X and Y domain, the dimension of the image. Here I used "0 to 50" in both x and y domain.
Use the output from the image sampler as values of the rotations.

The dimension of the wall should match then the dimension of the image if we want the image to fill the entire wall. To do so you should increase the number of grid cells in the RECGRID component in Ex and in Ey until the wall reaches the dimension of 50 in both x and y dimension (to match the domain values of the image) .
You can check the dimension of the wall by multiplying the spacing of the grid with the number of cells in both x and y direction






Untitled - Rhinoceros (Educational) - [Perspective]
File Edit View Curve Surface Solid Mesh Dimension Transform Tools Analyze Render V-Ray Help
Message: An entry with the same key already exists.
Command:
Perspective
B
X
Y
Z
Box
B
Cull
L
Pt
F
Z
V
G
A
C
X
Rot3D
G
X