FREEFORM GRID WITH VARIABLE HEIGHT

What you will learn

1. To rigorously handle complex geometry variation by mastering the use of tree data structure

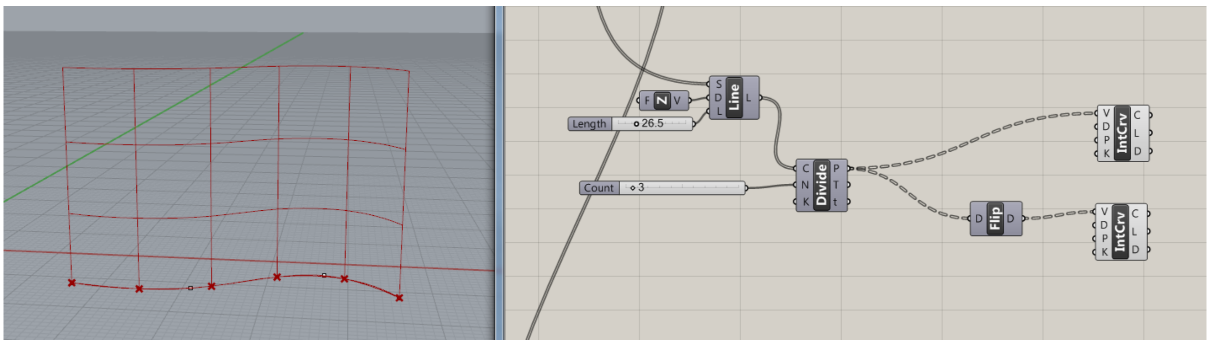
List of relevant components used

Graft Tree
Flip matrix
Sin/Cos
Evaluate
Interpolate Curve
Nurbs Curve
Loft

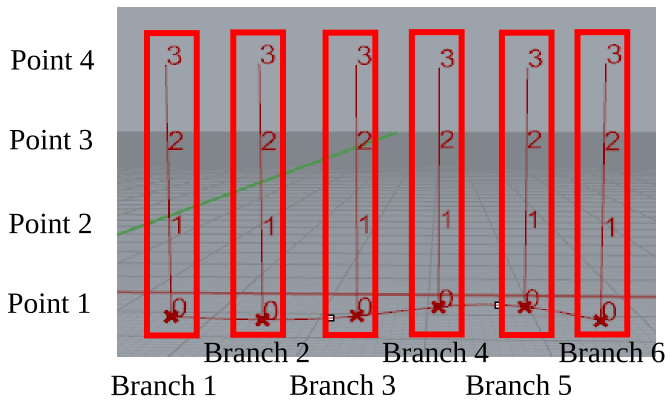
This exercise will get you through the procedures to create a freeform grid with variable extrusion height, by mastering the use of characteristic grasshopper tree data structure. The exercise will start from the step 12 of the previous exercise "Freeform Grid".

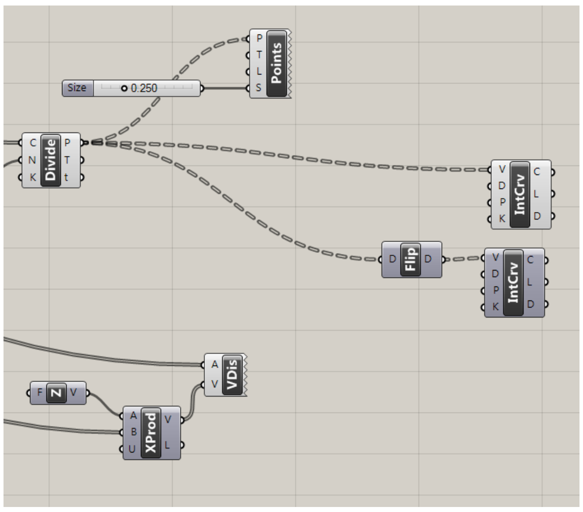
We´ll start from Step 12 of the exercise "freeform grid" of Grasshopper lecture 2, returned here below:

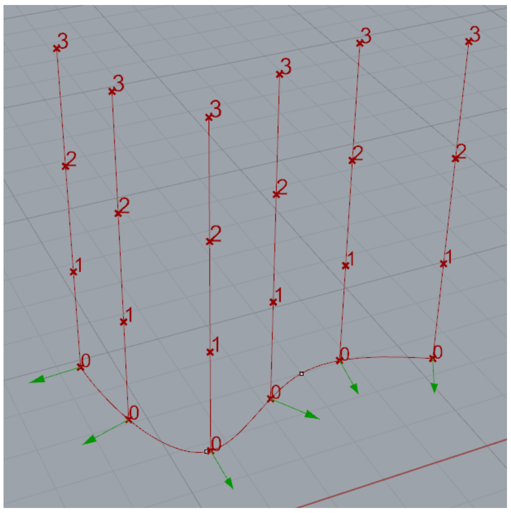
12 -You can remove the unnecessary components to have a cleaner canvas as below, and visualize the grid of curves.



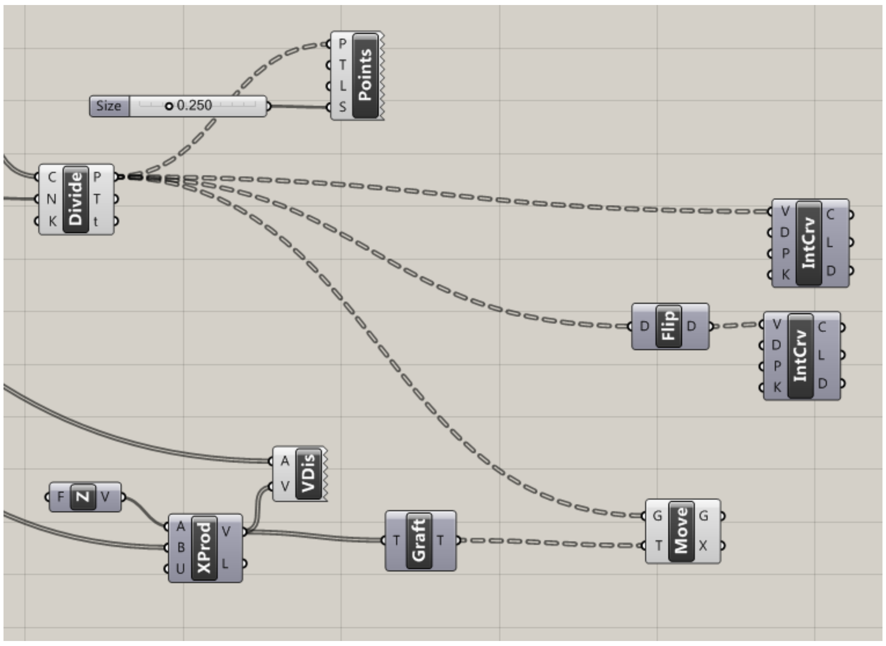
1. We want to move the points perpendicularly to the grid surface with a **move** component (transform>euclidean) and the perpendicular vectors created in step 7 of the grasshopper lecture 2 tutorial. Points are organized in branches, each branch contain a list of points in column (in this case 4 points), so all the points contained in a specific branch must move with the same perpendicular vector.

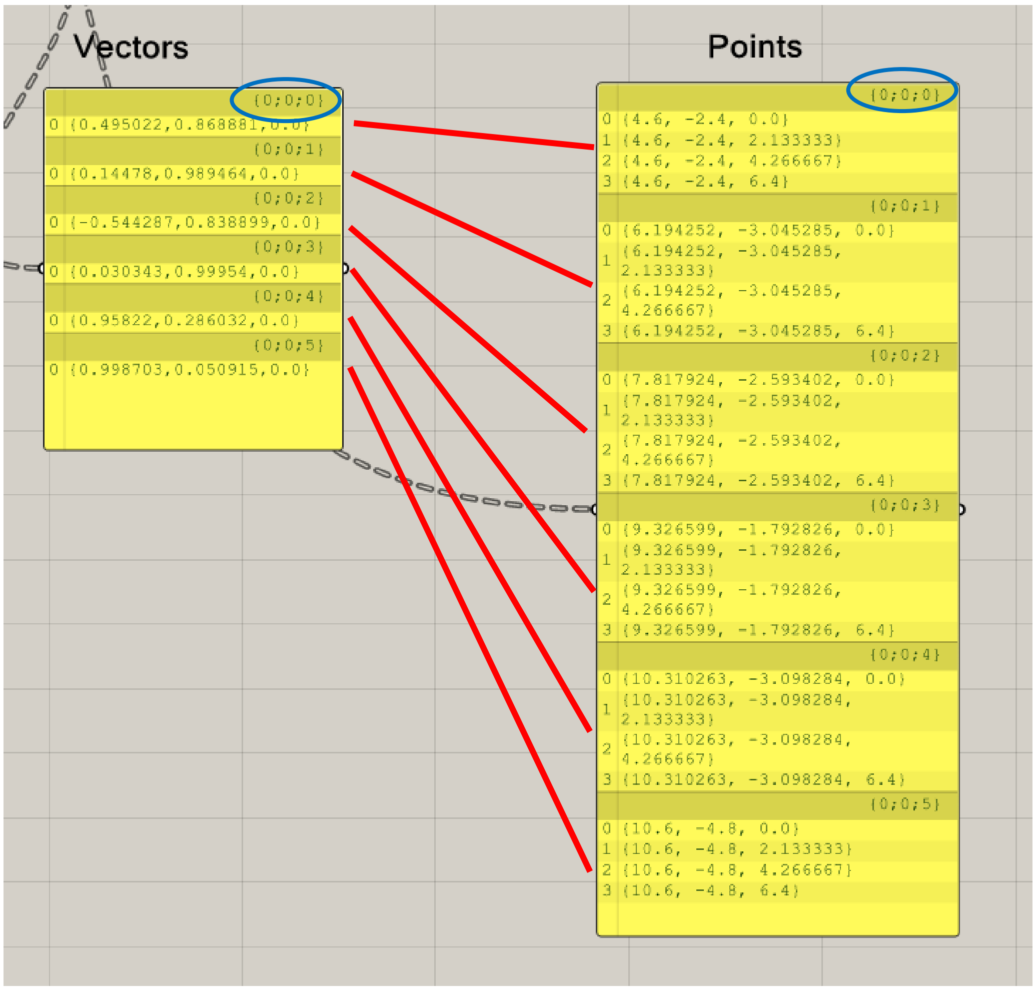






In order to associate each branch with the corresponding perpendicular vector, vectors should also be organized in branches. In this way the **move** component will displace the points in the first branch with the vector in the first branch. To store each vector into a different branch use a **graft** component. Vectors will be used to displace all the points contained in a branch with the same path. Vector in path {0;0;0} will move the points in branch of path {0;0;0}

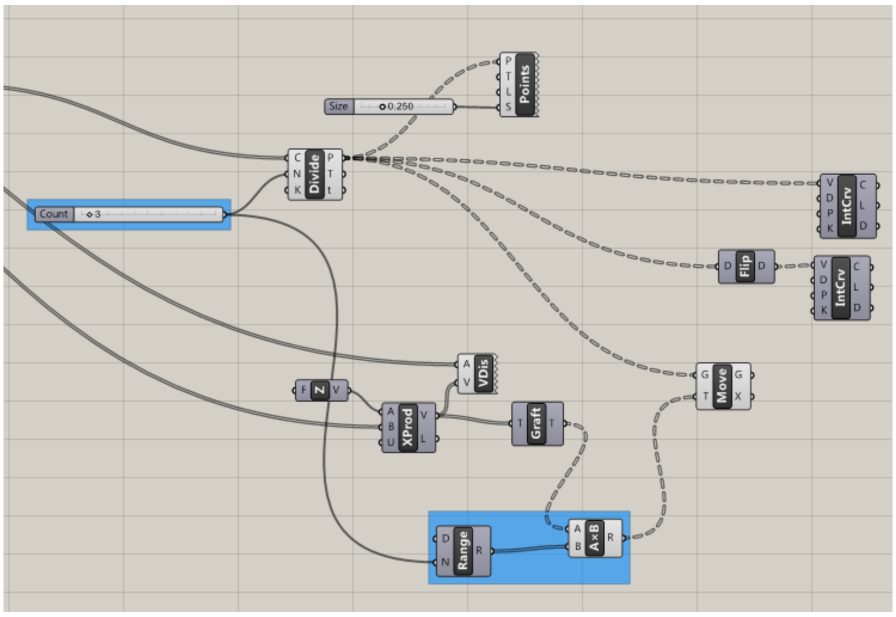




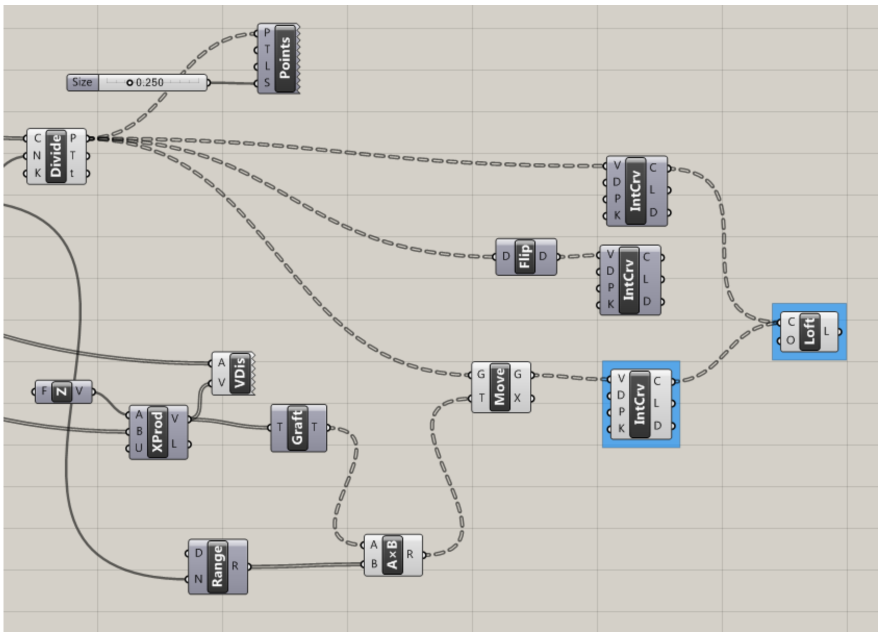
```
Vectors
                          {0;0;0}
0 {0.495022,0.868881,0.0}
                          {0;0;1}
0 {0.14478,0.989464,0.0}
                          {0;0;2}
0 {-0.544287,0.838899,0.0}
                          {0;0;3}
0 {0.030343,0.99954,0.0}
                          {0;0;4}
0 {0.95822,0.286032,0.0}
                          {0;0;5}
0 {0.998703,0.050915,0.0}
```

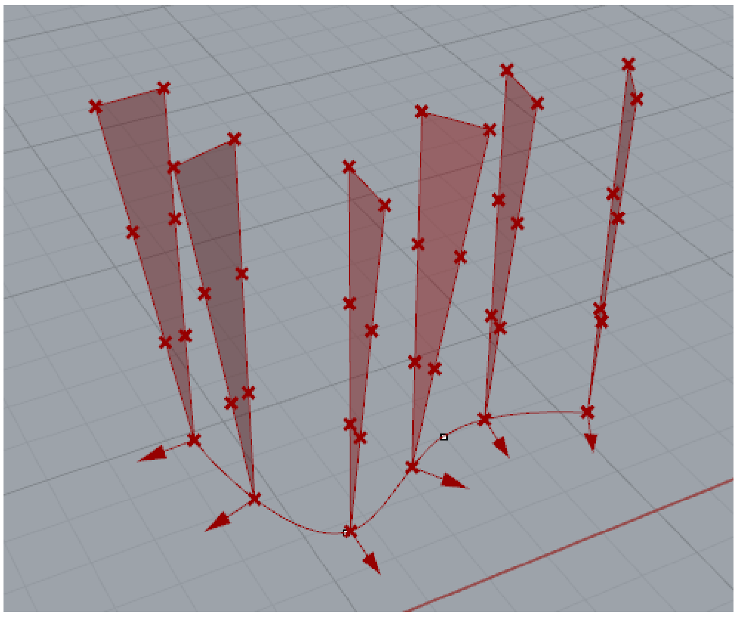
```
Points
                                     {0;0;0}
0 {4.6, -2.4, 0.0}
1 {4.6, -2.4, 2.133333}
2 {4.6, -2.4, 4.266667}
3 {4.6, -2.4, 6.4}
                                     {0;0;1}
0 {6.194252, -3.045285, 0.0}
1 {6.194252, -3.045285,
  2.133333}
2 {6.194252, -3.045285,
  4.266667}
3 {6.194252, -3.045285, 6.4}
                                     {0;0;2}
0 {7.817924, -2.593402, 0.0}
1 {7.817924, -2.593402,
  2.133333}
2 {7.817924, -2.593402,
  4.266667}
3 {7.817924, -2.593402, 6.4}
                                     {0;0;3}
0 {9.326599, -1.792826, 0.0}
1 {9.326599, -1.792826,
  2.133333}
2 {9.326599, -1.792826,
  4.266667}
3 {9.326599, -1.792826, 6.4}
                                     {0;0;4}
0 {10.310263, -3.098284, 0.0}
1 {10.310263, -3.098284,
  2.133333}
2 {10.310263, -3.098284,
  4.266667}
3 {10.310263, -3.098284, 6.4}
                                     {0;0;5}
0 {10.6, -4.8, 0.0}
1 {10.6, -4.8, 2.133333}
2 {10.6, -4.8, 4.266667}
3 {10.6, -4.8, 6.4}
```

2. If one wants to move the four points of a specific branch to move differently one to another, one should create in the correspondent vector branch one vector for each of the four point. The suggested method here is to multiply the vector of each branch with a series of 4 numbers or factors (i.e a series of numbers that matches the number of points). In each vector branch four different vectors will be generated with the same direction but with a different length, determined by the factors. To create the list of numbers/factors one can use for example the range component and use in N the same input used to create the number of points, in order to have always matching lists of points and numbers (and consequently vectors).

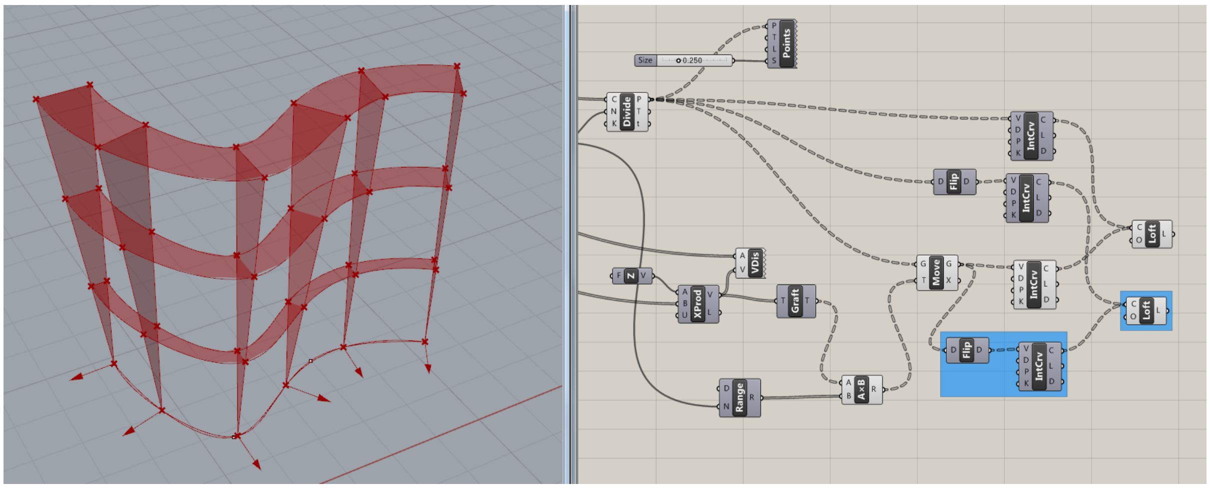


3. Now we create curves with the displaced points with Interpolate (Curve>Spline) and create surfaces between the original curves and the displaced ones with the **loft** component (**surface>freeform**)



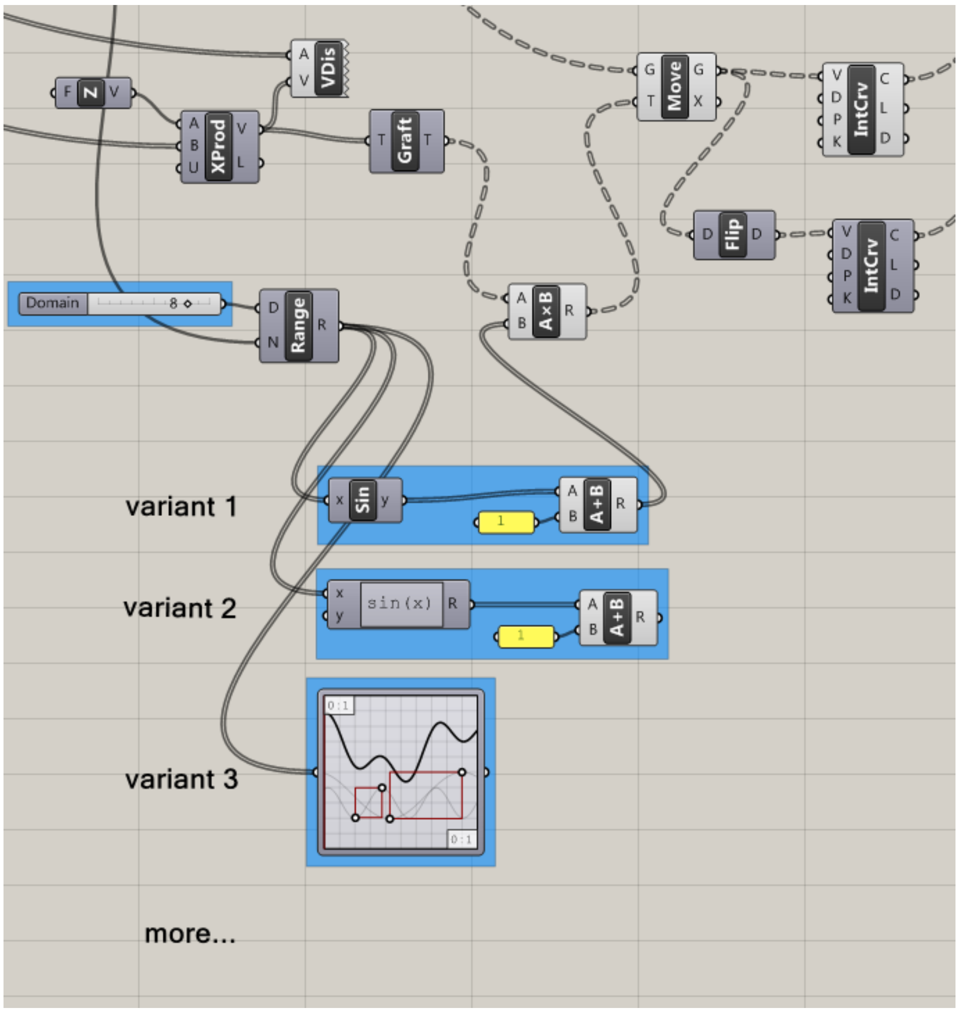


4. Let´s create the curves in the other direction after flipping the data structure of the displaced points, and then loft with the correspondent original curves. The geometries in the two loft components constitute your final freeform geometry. By moving the four original points in Rhino, the 3d grid will adapt and modify in real time.



5. Now experiment with different factors: instead of a simple sequence of numbers obtained with the “Range” component alone, use other methods to create list of numbers and consequently to explore different resulting geometries. One example is to interpose a **Sine** (Maths>Trig) component to generate sine waves geometries. Depending on the portion of the sine function one wants to recreate, it could be necessary to modify the D input in the Range component. Because sine output can range from -1 to 1, in order to have always positive values one should add 1 to the output. The same sine function can be created with the component **Expression** (Math>script): this component however allows you to define more complex expressions.

6. Explore with this and other ways to generate geometries with different factors, f.ex with "graph mapper"



7. What would happen if you used **Nurbs Curve** components (Curve>Spline) instead of Interpolate (IntCrv) to generate the grid geometry? Would the resulting geometry still be usable? Why?