

# **Parametric modelling**

## **Grasshopper basics tutorial**

## Abstract

Author(s) Pavel Aksenov	Publication type Bachelor's thesis	Completion year 2022
	Number of pages 100	
Title of the thesis <b>Parametric modelling</b> Grasshopper basics tutorial		
Degree Bachelor of Civil and Construction Engineering		
Name, title and organisation of the client LAB University of Applied Sciences		
Abstract <p>This thesis was performed for LAB University of Applied Sciences to create educational material for BIM engineers.</p> <p>This thesis describes the idea of parametric modelling, shows the use and implementation cases of parametric modelling, and shows parametric modelling software.</p> <p>Explains concepts such as node modelling, types of parametrization, parameters, components, math applications, data structures, etc.</p> <p>The main goal of this thesis is to present brief material, required to understand how to work in software like Grasshopper, Dynamo, etc.</p> <p>The appendix is a step-by-step course of Grasshopper, made for the BIM-ICE Project.</p>		
Keywords Parametric modelling, Grasshopper, Rhinoceros, BIM		

## Contents

1	Introduction.....	1
2	How Parametric modelling works.....	2
2.1	Use cases and examples.....	2
2.2	Node programming.....	3
2.3	Types of parametrization.....	4
2.3.1	Tabular parametrization.....	4
2.3.2	Variational parametrization.....	4
2.3.3	Geometric parametrization.....	4
2.4	Software.....	5
3	Theory.....	6
3.1	Base parameters.....	6
3.2	Primitive parameters.....	8
3.2.1	Integers.....	8
3.2.2	Numbers.....	8
3.2.3	Boolean.....	9
3.2.4	Color.....	10
3.2.5	Text.....	10
3.3	Components.....	11
3.3.1	Point.....	12
3.3.2	Vector.....	13
3.3.3	Curves.....	14
3.3.4	Surface and brep.....	15
3.4	Math.....	16
3.4.1	Domain.....	16
3.4.2	Trigonometry.....	17
3.5	List.....	17
3.5.1	Tree.....	17
3.5.2	Sequences.....	18
3.5.3	Interactions with list.....	19
3.6	C# in Grasshopper.....	20
4	Summary.....	23
	References.....	24

## Appendices

Appendix 1. The Basics of Parametric Modelling (Course material)

## Abbreviations

BIM – Building Information modelling

CAD – Computer Aided Design

IT – Information Technology

NURBS - Non-uniform Rational Basis Splines

IFC - Industry Foundation Classes

## 1 Introduction

In recent years BIM modelling became the most popular approach in building design. It allows the creation of a solid model that contains all information about the entire project. Unlike CAD modeling, BIM provides more automatization for the designing process, which leads to a reduction in inconsistencies in a project.

Parametric modelling is a BIM tool, that creates possibilities for applying even more automatization for the designing process. This approach is significantly different from the direct modelling process. In the case of parametric design, it creates a mathematical model of objects with parameters (The Dynamo primer for Dynamo v 2.0, n.d.). Changing parameters leads to changes of model configuration, mutual movements of objects, etc.

The process of parametric model creation takes more effort than direct modeling, but it gives much more opportunities when difficult geometry is required. Parametric model is dynamic, which allows to change any objects, connected with parameters. This model can be used in all project stages: from shaping to production.

The first goal of this thesis is to show parametric modelling tools for BIM engineers. The second goal is to teach users without IT background how to use Grasshopper and implement it in the projects. It will be especially useful for architects.

## 2 How Parametric modelling works

### 2.1 Use cases and examples

Two general use cases for parametric modelling in BIM are:

- Complex geometry creation
- Generative design.

It is almost impossible to create models of some architectural concepts with tools that BIM software can provide. In this situation, a designer can rather wait for updates in his software or create a plug-in by himself. This is when parametric modelling can be useful since it gives the opportunity to create required instruments without deep knowledge of IT.



Figure 1. Beijing National Aquatics Center (Adobe stock) (Source)

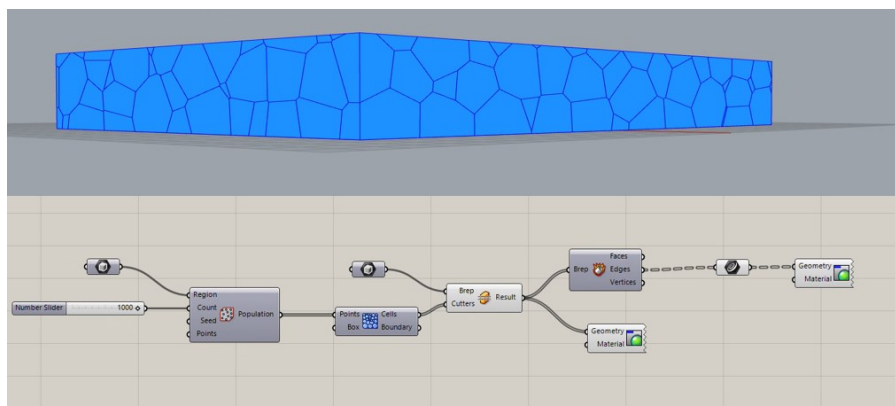


Figure 2. Geometry concept of Beijing National Aquatics Center, done in Grasshopper

The second use case is a Generative design. The Generative design is the approach that uses the method of multiple iterations to find the solution for the initial problem. For example, it is possible to create an algorithm that will search for the best location for a building by considering such factors as insolation, viewpoints, shading, etc.

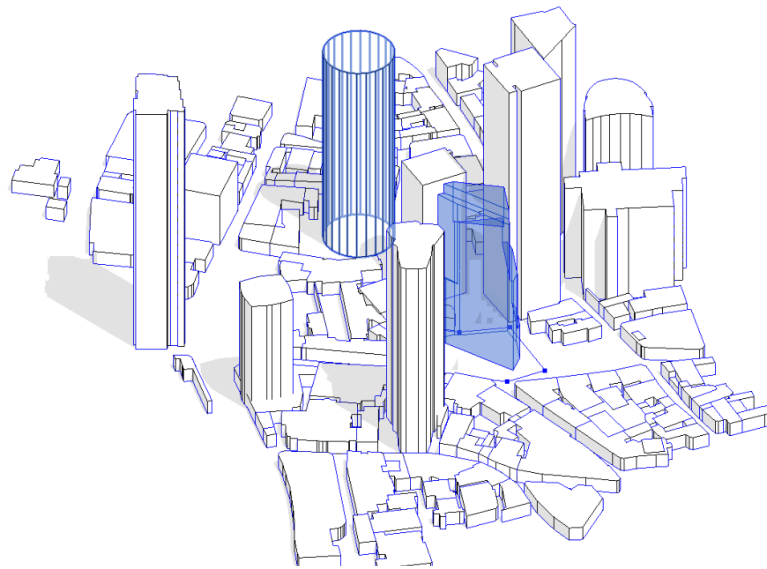


Figure 3. Generative design model (Seiler Design Solutions) (Source)

## 2.2 Node programming

In programs like Dynamo and Grasshopper parametric models can be created with high-level programming language based on nodes. Unlike Python, C++, or many other programming languages, Dynamo and Grasshopper do not use written code in process of script creation, but use node system, which allows to create script by using nodes.

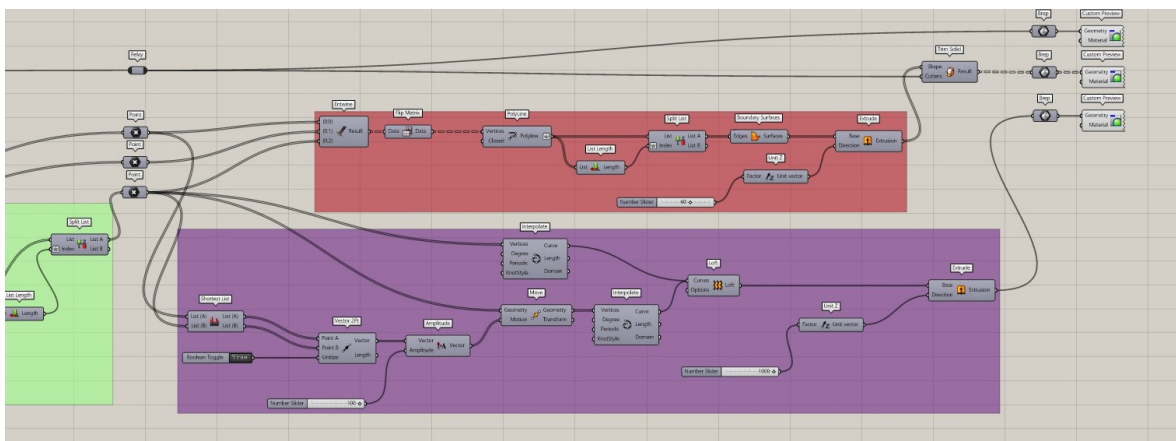


Figure 4. Example of the node algorithm in Grasshopper

Node (method or function) is a pre-made command input slot, body, and output slot. Input slots are used to take data, such as numbers, points, lines, etc. The body is a function that manipulations with data. And the output is a slot, used to translate this new data to other nodes.



## 2.3 Types of parametrization

### 2.3.1 Tabular parametrization

Tabular parametrization consists of creating a table of parameters of typical objects. Creation of a new object exemplar is done through choosing from table of object sizes. The possibilities of tabular parametrization are very limited, since the assignment of the new values of parameters and geometric dependences is usually impossible (Wikipedia, Parametric modelling, 2022.)

However, tabular parameterization is widely used in all parametric BIM systems, since it can significantly simplify and speed up the creation of libraries of standard and typical objects (Wikipedia, Parametric modelling, 2022.)

### 2.3.2 Variational parametrization

Variational or dimensional, parametrization is based on the construction of sketches (with the imposition of various parametric dependences on the sketch objects) and the imposition of restrictions by the user in the form of a system of equations that determine the dependencies between the parameters. Variational parametrization makes it easy to change the shape of a sketch or the value of operation parameters, which makes it convenient to modify a 3D model (Wikipedia, Parametric modelling, 2022.)

### 2.3.3 Geometric parametrization

Geometric parameterization is a type of parametrization, in which the geometry of each parametric object is recalculated depending on the position of parent objects, their parameters, and variables.

A parametric model, in the case of geometric parametrization, consists of building elements and image elements. Building elements define parametric dependencies. Graphic elements include graphic models, as well as design elements (dimensions, inscriptions, hatching, etc.).

Some building elements may depend on other building elements. Building elements can also contain parameters (for example, the radius of a circle or the angle of inclination of a straight line). When one of the elements of the model is changed, all elements dependent on it are rebuilt following their parameters and ways of setting them (Wikipedia, Parametric modelling, 2022.)

## 2.4 Software

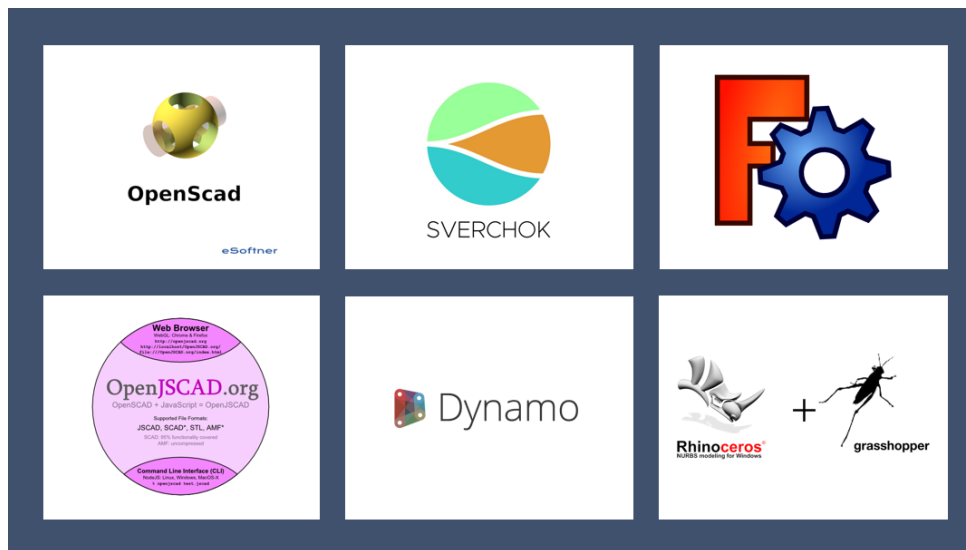


Figure 5. Parametric modelling software

Today there is a bid variety of parametric software, such as Dynamo for Revit, Grasshopper for Rhino, or free software, such as SVERCHOK for Blender, OpenSCAD, FreeCAD, or browser apps such as JSCAD. All this software is used for different problems, and sometimes, some solutions require a combination of programs.

Dynamo is a great tool for Revit. The biggest advantage is integration into Revit. Dynamo allows to work with most of Revit objects and their parameters, but the biggest disadvantage is an incompetent other BIM software.

Grasshopper is a plug-in for Rhinoceros. The focus of Grasshopper is geometry since it uses a mesh geometry system. Grasshopper, unlike Dynamo, can be integrated into a bigger amount of BIM soft, such as Tekla Structures, ArchiCAD, and Revit, but it solves other problems.

FreeCAD and OpenSCAD have no integration possibilities, but they can be used to work with IFC files, which can be imported from every modern BIM software. These programs are very useful since they are free.

JSCAD is a browser app that provides the creation of parametric models by using JavaScript. This app can be helpful when an API connection is required.

### 3 Theory

#### 3.1 Base parameters

Grasshopper nodes are divided into two main groups:

- Parameters - a group of nodes, used to collect, or convert information. Parameters contain numbers, text, colors, geometry, bool values, file links, etc.
- Components work with parameters and convert them in different ways. Components are usually logically divided into thematic sections. Each section is characterized by a common idea, for example math, curves, vectors, groups, etc.

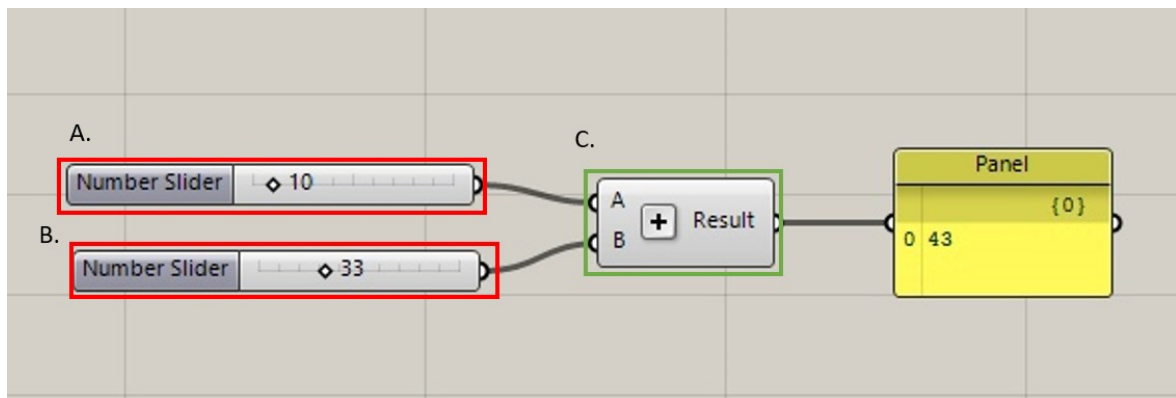


Figure 6. Example of node working process in Grasshopper

This is an example of two numbers addition (figure 6). Nodes A and B are number parameters, where node C is a component of addition.

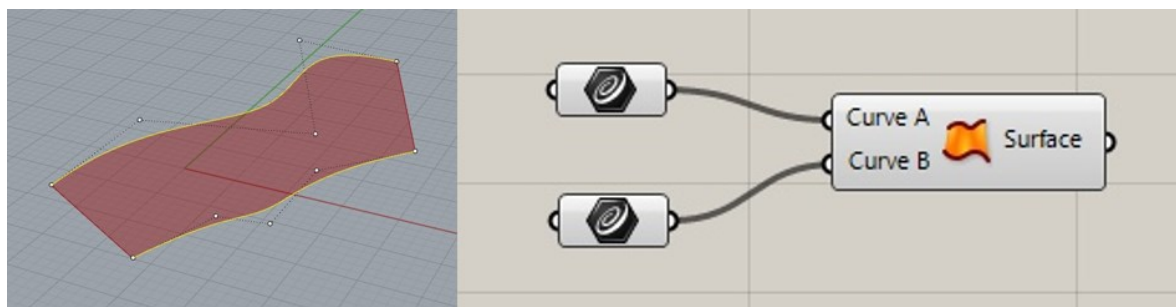


Figure 7. Process of NURBS curve creation

On this example, two curves were taken from Rhino to make a surface between them. Even though there are components that can be used to create curves, in this particular case, these elements will be considered as parameters, where node "Ruled surface" is a component.

Parameters in Grasshopper can be divided into three groups:

- Input - are used to create different values inside Grasshopper, straight on canvas, or can be used as an external input. These nodes can be found in the “Input” section
- Primitive - parameters are used for basic values, such as numbers, text etc.
- Geometry - Where geometry parameters are used to create/take elements inside/from a Rhinoceros.

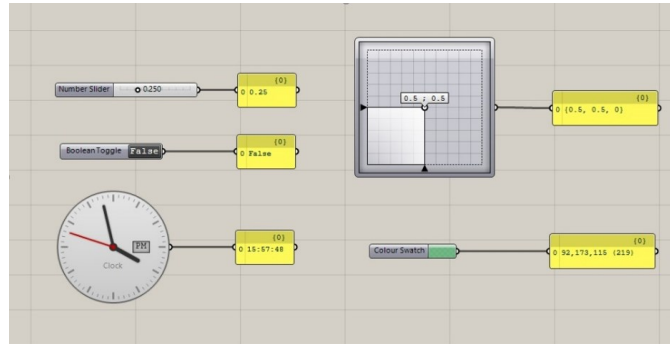


Figure 8. Example of input nodes

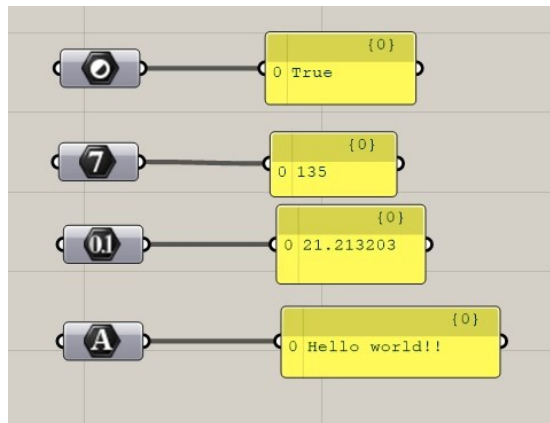


Figure 9. Basic primitives in Grasshopper



Figure 10. Geometry parameters in Grasshopper

## 3.2 Primitive parameters

Primitive parameters allow user to create different data types, to convert one data type to another, to link file from PC to Grasshopper and to use files from Rhino (for example “Shaders”) etc.

Primitives can be divided into three groups:

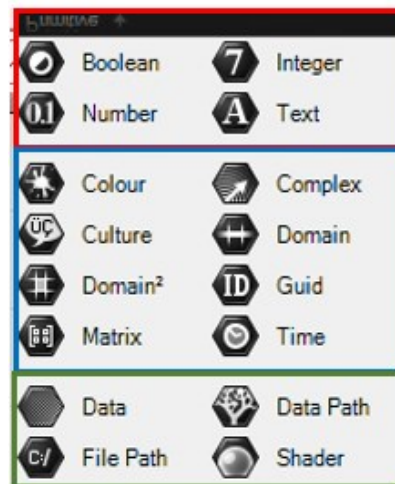


Figure 11. Primitive parameters division in Grasshopper

- First group are number, boolean (bool) and text parameters. Those Parameters are used to set numbers, bools, or text and for converting. This operation could be helpful in different situations. For example, it can be used to convert bool values into numbers and vice versa
- Second group of primitives contains methods, used for more complex data, such as colors, number domains, complex numbers, matrixes, time end etc.
- Third group is used to work with file links.

### 3.2.1 Integers

Primitive Integer is used for whole numbers creation, lists of whole numbers, or to convert other data types into integers. Where whole numbers – are natural numbers without numbers with coma and including zero and all negative natural numbers.

### 3.2.2 Numbers

Primitive “Number” can be used to create numbers with floating points (ex. 1.15; -4.654; 2.88). Also, it can be used to convert data into float numbers. Those numbers have a certain

number of digits after the coma. Grasshopper uses periods in floating numbers due to language peculiarities.

### 3.2.3 Boolean

Primitive Boolean allows the creation of both a single value and a list of values. It is a commonly used data type in Grasshopper. It contains only two values: True and False. These values correspond to two values “1” and “0”. In other words, True = 1 and False = 0. More broadly, the False value means the absence of something, where True means the presence of anything. It means that can be corresponded not only with one, but also with any number except zero, or any geometry, such as a point, vector, etc.

Bool list (or bool pattern) can be used as a bool mask for filtering values from the list. One of the most common examples of bool mask is the list where even elements are True and odd elements are False (or vice versa). This bool pattern can be used to take every second element from the list.

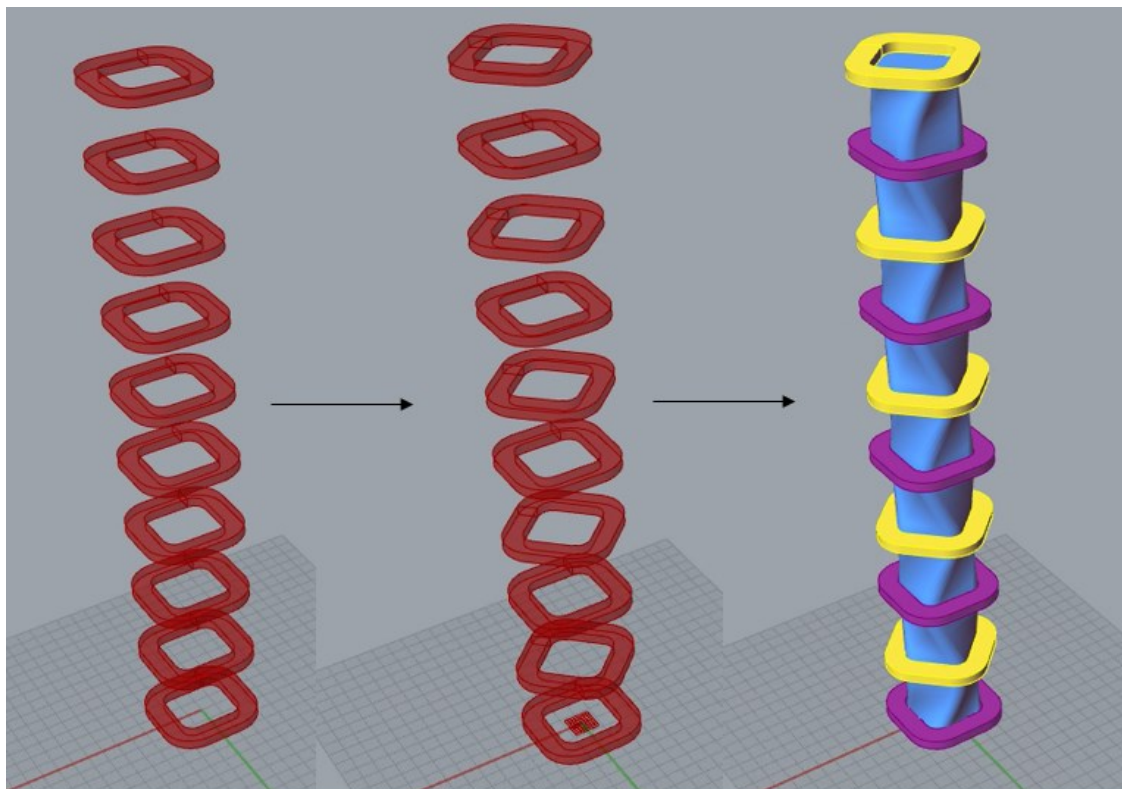


Figure 12. Workflow of creating parametric tower

### 3.2.4 Color

There are a lot of possible nodes to create color in Grasshopper, however, output data from each of them is always the same. The color can be defined by four numbers (ARGB system). Each number has a value between zero and 255. The first number – is the amount of red color, the second is the amount of green, the third is the amount of blue and the fourth number, which is in brackets and divided by space from the third one is the alfa channel. If A equals 255, it will not appear in a result.

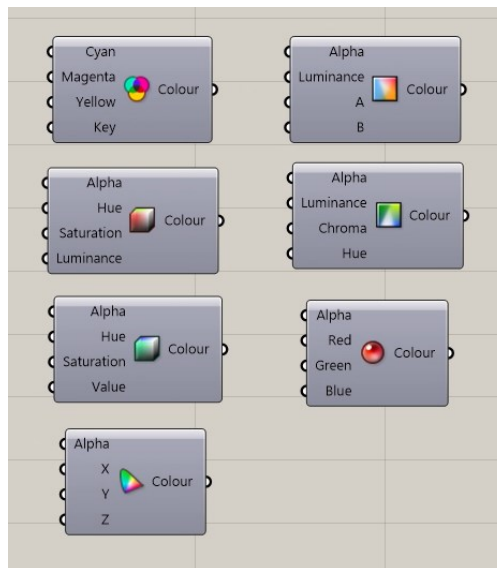


Figure 13. Nodes for color creation in Grasshopper

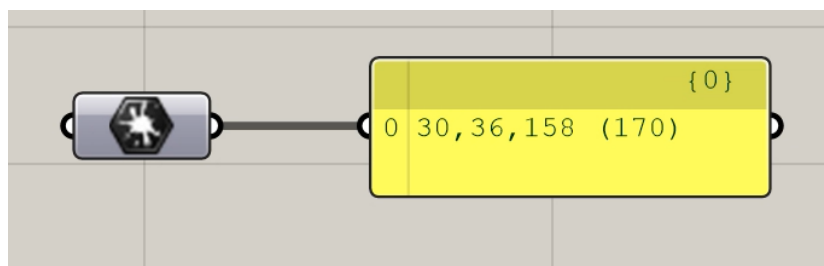


Figure 14. Color output format

### 3.2.5 Text

Grasshopper text is the most complex parameter since in different contexts it can be used as any other primitive. By specifying text values with syntactically correspond to other data types, methods will take the values as appropriate. For example, by giving value “True” for the text node, it can be used for the “And” operator as a bool value, by giving a number value, the text will be considered as a number, by typing three numbers, divided by coma,

text can be considered as a color, etc. This concept opens new possibilities for working with strings through text-based programming languages, which will be discussed later.

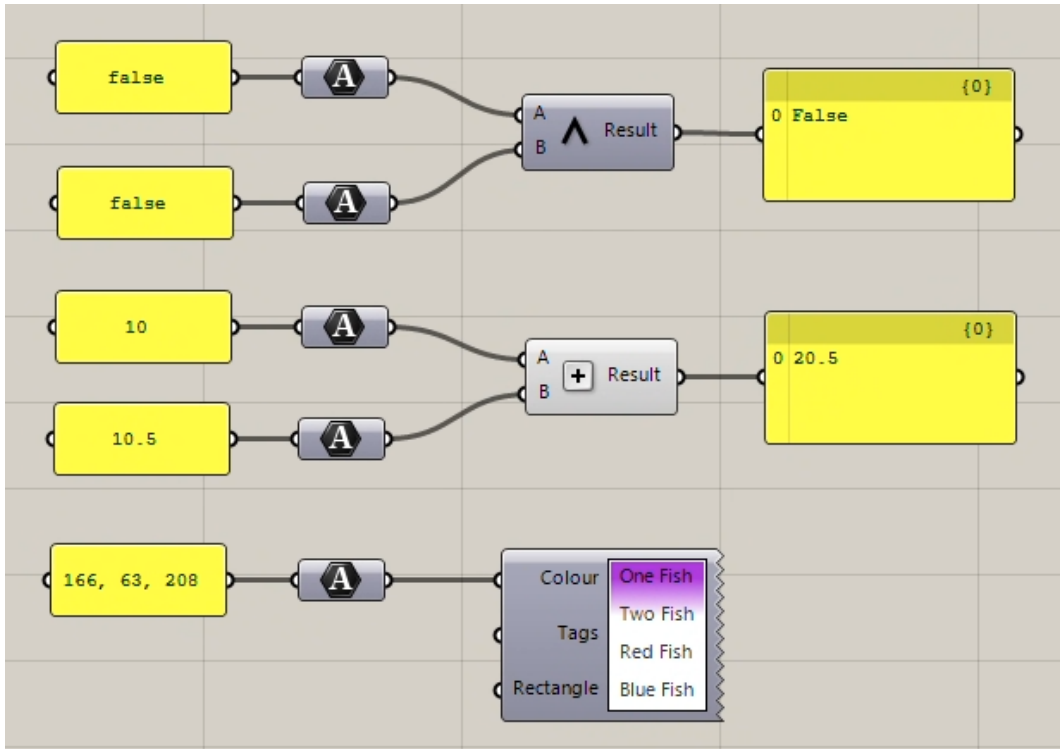


Figure 15. Text data conversion examples

### 3.3 Components

Components are used for:

- geometry creation
- data display
- manipulation with existing geometry
- manipulation with existing data.

To make the component work, it requires input data. Primitive parameters or another component can be used. Most components require certain data for input since negative numbers cannot be used for the count, bool cannot be used for coordinates, the surface cannot be used for math operations, etc. Also, some methods have limited values for input, for example, colors are a combination of three values between zero and 255. Some components can work only with a list or data tree.



### 3.3.1 Point

Point is the simplest geometry object. A lot of geometry requires a point as a base or reference object. Points in 3D space have three coordinates, usually denoted as  $[x, y, z]$ . Points in two-dimensional space have two coordinates. That coordinates can be called either  $[x, y]$  or  $[u, v]$  depending on which two-dimensional space is considered. Two-dimensional space is a continuous coordinate system, that strives for infinity. It means that coordinates  $x$  and  $y$  can have values from minus infinity to plus infinity the only limitation – is computer power, where  $u$  and  $v$  coordinates show the point location on the surface. The surface can have any dimensions, but the UV coordination system will always be limited between zero and one for each dimension.

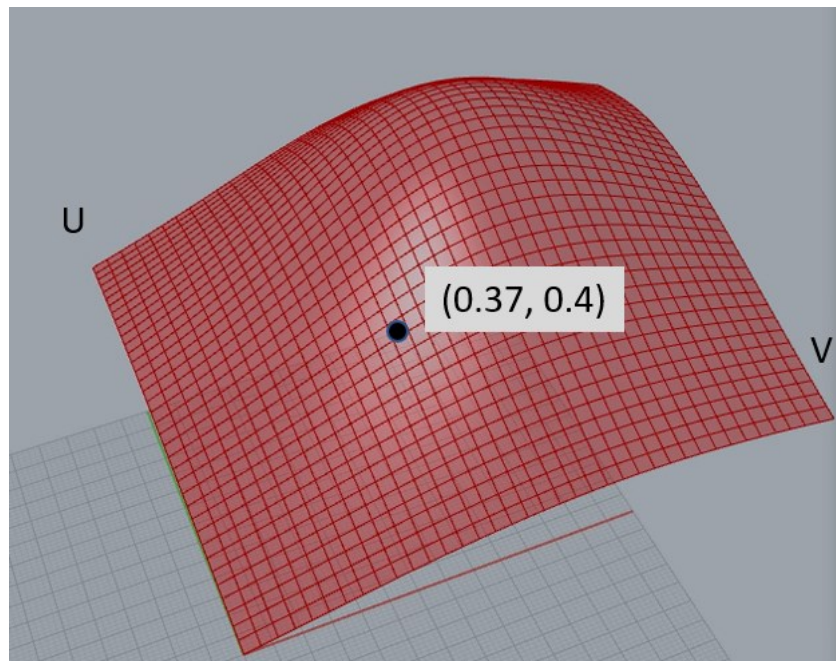


Figure 16. Point in UV coordinates of the surface

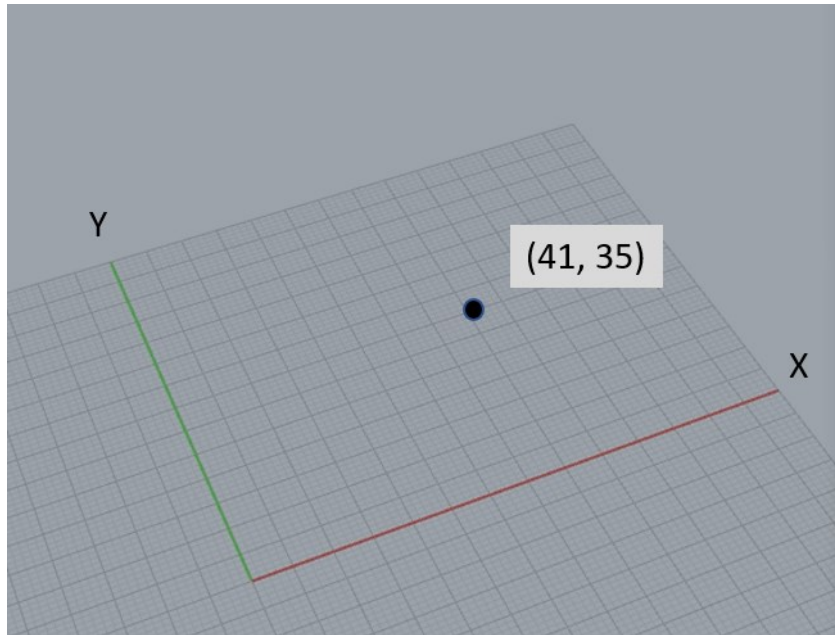


Figure 17. Point Cartesian 2D coordinates

### 3.3.2 Vector

Vector is an abstract geometry that helps to define the location, orientation, and special context for other geometry (The Dynamo primer for Dynamo v 2.0, n.d.). Vector is a geometric quantity describing direction and magnitude. Vector like a point can be defined by three numbers. Unlike point, vector has no location, it has only direction and length. Since vector is an abstract geometry, it cannot be drawn, but it is possible to draw the line that will represent this vector in certain coordinates.

The numbers defining the vector indicate the coordinates of the point that must be connected to the origin by an abstract line. The length and direction of this line are equal to the length and direction of the vector.

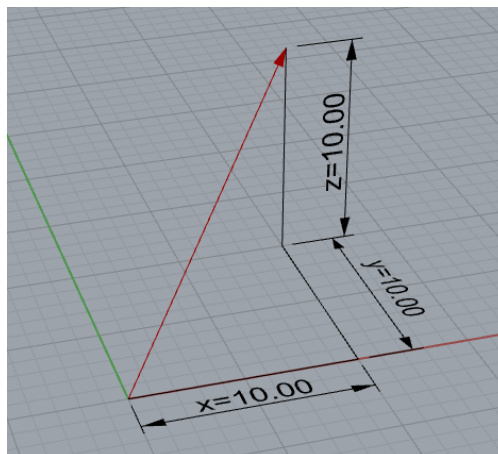


Figure 18. Non-abstract vector representation

Vectors with a length equal to one are called utilized. Coordinates of a utilized vector are always laying on a sphere with a radius equal to one and origin in (0, 0, 0) coordinates.

### 3.3.3 Curves

Curves are a visual representation of mathematical functions with one argument. Some curves, like lines or circles, have simple mathematical justification, whereas Polycurve is a more complex object and in most cases cannot be defined by a simple function. That is why it is easier to determine curves like a set of points, connected by different connections.

Curves like points are intermediate geometry, used for the creation of more complex forms. There are seven general types of curves:

- Line – is a curve between two points
- Polyline – is a curve, created by joining two or more lines
- Arc – is a segment of the circle, that requires origin, radius, and angle
- Circle – is a closed curve with a round shape, that has a radius and origin
- Ellipse – is a closed curve, that has an origin and two radiuses: the first radius is the width of a narrow part, and the second is the width of a wide part
- NURBS - (Non-uniform Rational Basis Splines) are mathematical representations that can accurately model any shape from a simple two-dimensional Line, Circle, Arc, or Rectangle to the most complex three-dimensional free-form organic Curve. Because of their flexibility (relatively few control points, yet smooth interpolation based on Degree settings) and precision (bound by a robust math), NURBS models can be used in any process from illustration and animation to manufacturing (The Dynamo primer for Dynamo v 2.0, n.d..)
- Polycurve is a combination of any curves from above.

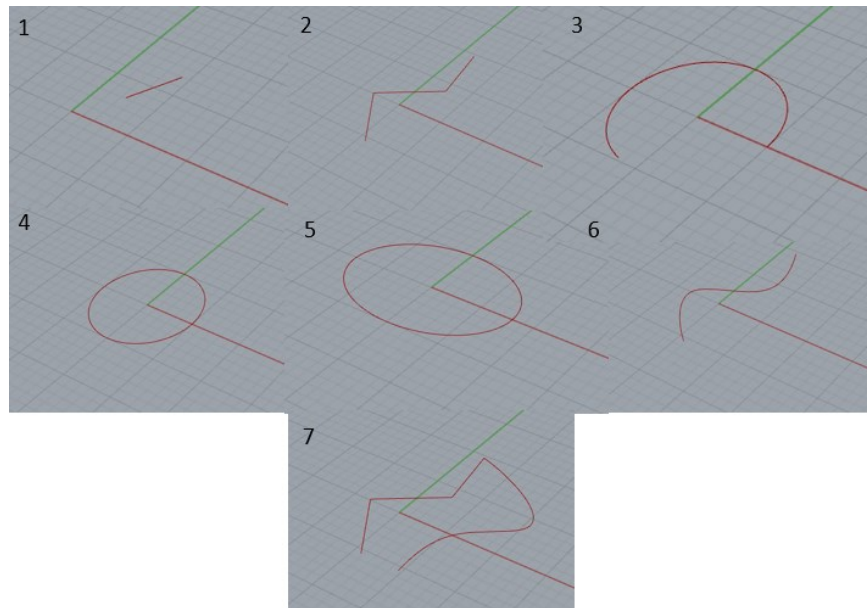


Figure 19. Curve types

### 3.3.4 Surface and brep

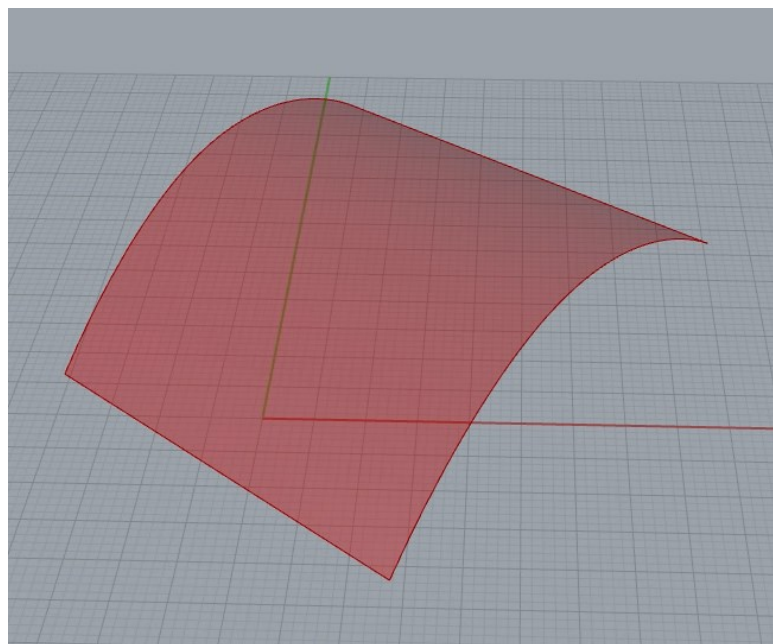


Figure 20. Example of a surface

The surface is a representation of a mathematical representation of a function and two parameters. Surface parameters are called “U” and “V”, they are used to define segments. The surface can have only two sizes, so it can be explained as an infinitely thin shell.

Where brep is every object that has two or three sizes, such as cylinders, spheres, etc. so the surface can be considered as a special case of brep.

### 3.4 Math

Pre-made components are not always the best tools for certain goals. But to get the desired result, math functions can be used to modify datasets and use them for geometry creation. A simple example of a math function is a Sine, the coordinates of which can be found by the formula:  $Y=a*\sin (bx+c)$ , where (a) - is an amplitude, (b) – is a multiplication phase shift, (c) – is a phase shift, and (x) – is an argument.

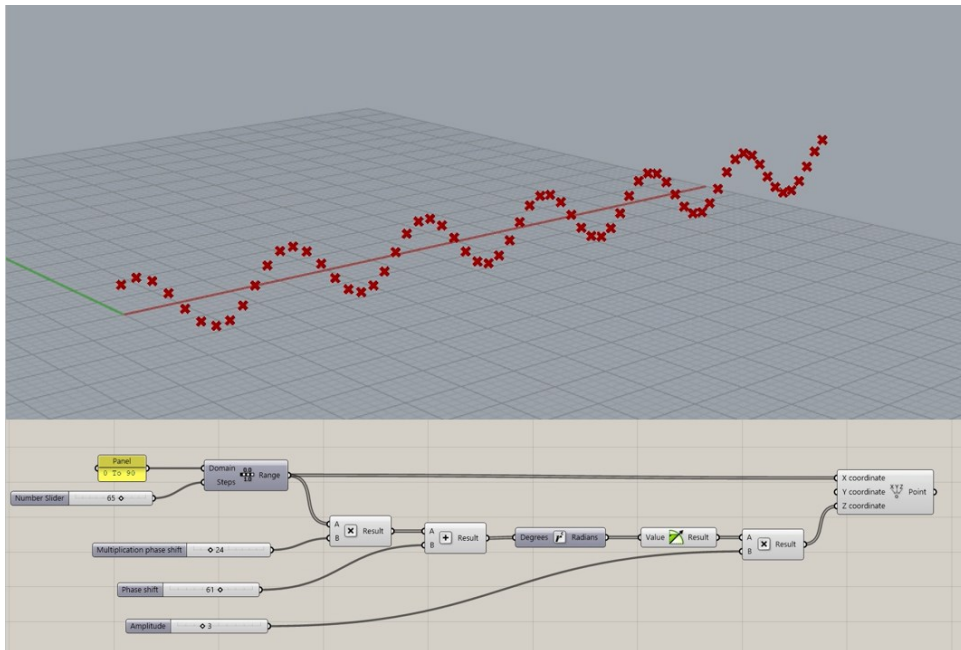


Figure 21. Algorithm of SINE line creation

#### 3.4.1 Domain

A domain is a numeric area between the lower and upper limits. Domain does not generate numbers by itself, but it can be divided into numbers by certain rules, to create number sequences. There are an unlimited number of real numbers between boundaries. A domain can be written as “X To Y”, where X is a lower limit and Y is an upper limit.

The 2D domain is a combination of two domains, represented by U minimum, U maximum, V minimum, and V maximum values. These kinds of a domain are used for surface geometry, to dispatch segments.

### 3.4.2 Trigonometry

Trigonometry is a widely used concept in parametric modelling since Sine, Cosine, and Tangent functions are the foundation for lots of concepts, based on rotation. These functions allow the creation of trigonometric number sequences, that are used as coordinates in advance.

Trigonometry functions in Grasshopper use radians for input slots. Usually, degrees are more comfortable to use for angles, and to do so, conversion functions should be used. If parametric modeling software does not have this function in the library, the formula " $A = \text{Arad} * 180 / \pi$ " should be used.

### 3.5 List

Any object in Grasshopper is a set of data, structured in a certain way. It can be structured as a single piece of data, list, or tree. In Grasshopper single piece of data is a list with a single item.

A list of data in Grasshopper is used to simplify algorithms with multiple objects. It is much more convenient to create a range of points with few functions and receive a list of points than to create every single point manually. Also, nodes like "Interpolate" can work only with non-empty and non-single item lists. In other words, it requires a list with at least two points. In a list, each element has an index, and the numeration of indexes always starts with zero.

#### 3.5.1 Tree

A hierarchical structure for storing data in nested lists is called a data tree. Data trees should be created when components are structured to take a set of data as input and convert it into multiple datasets. Grasshopper works with new datasets by nesting these sets into sub-lists. Each sub-list can contain another sub-list, so the data tree has the property, that represents a number of nesting.

For navigation purposes, each sub-list and each element inside a data tree has an index. The standard view of this index is "{0, a, b, ... n}", where "a" represents an index of the sub-list in the main list, "b" shows an index of the sub-list in the list "a" etc.

### 3.5.2 Sequences

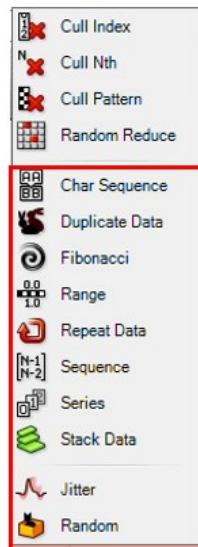
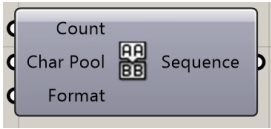

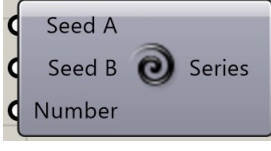
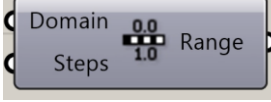
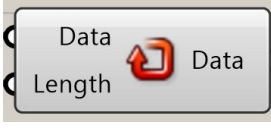
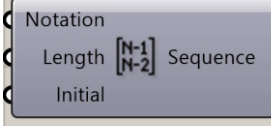


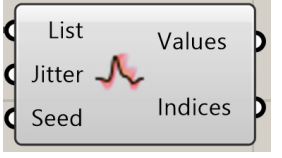
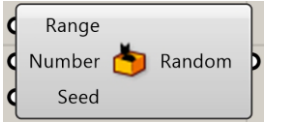


Figure 22. Sequence nodes in Grasshopper


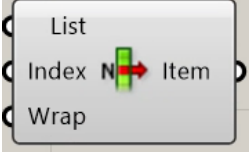
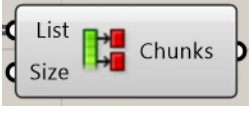

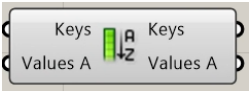
Table 1. Sequence nodes in Grasshopper

Char sequence		Creates the sequence of char symbols
Duplicate data		Duplicates data required number of times
Fibonacci		Generates Fibonacci sequence by giving first and second items and the length of the series.
Range		Divides domain to numbers required number of times
Repeat data		Repeats data in the list till it gets to required length.
Sequence		Generates a sequence of number according to the formula till it reaches required length

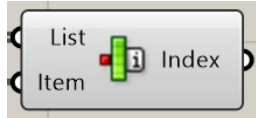
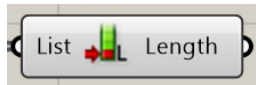

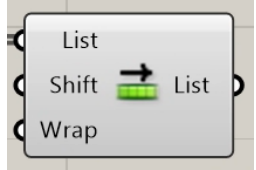
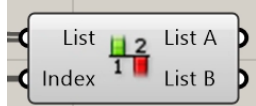
Series		Generate a series of numbers by addition step to previous number till it reaches required count.
Stack data		Takes two lists for input. First list with data and second with multiplication numbers and duplicate data according to the second list values.
Jitter		Randomly shuffles list values
Random		Generate a list of pseudo random numbers (according to a seed)

### 3.5.3 Interactions with list

Table 2. Nodes for interactions with a list

Insert Items		Put collection of items into list by certain indexes.
List Item		Take item (or items) using indexes from the list
Partition List		Creates Sub-lists from one list
Reverse List		Read list from the end to beginning
Sort List		Sort listed values from $-\infty$ to $\infty$ and from "A" to "Z"



Item Index		Searches index of a certain item
List Length		Counts elements of the list
Replace Items		Replace elements in list by indexes
Shift List		Offset all elements in a list
Split List		Creates two lists from one

### 3.6 C# in Grasshopper

Grasshopper supports C# scripts, they can be done through the node called “C# Script”. It is a strong instrument to work with.

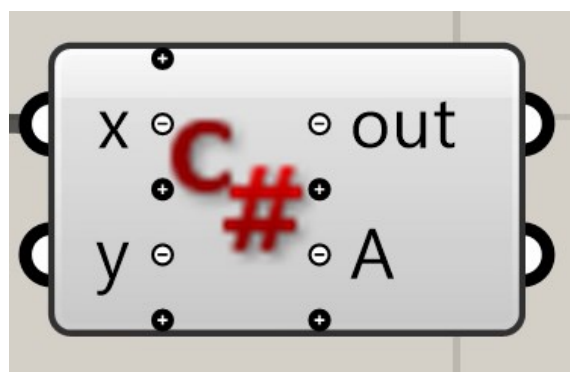


Figure 23. C# node in Grasshopper

By default, this node has two input and two output slots. The number of slots can be regulated through “plus” and “minus” buttons. All input and output slots must have names, according to the C# variable naming rule since they will be used as variables in a script in advance.

“Out” output is a special value, that can be declared through two pre-written methods inside the node:

- Print() for strings
- Reflect() for geometry

These two functions are written in Utility functions block

```

public class Script_Instance : GH_ScriptInstance
{
    #region Utility functions
    /// <summary>Print a String to the [Out] Parameter of the Script component.</summary>
    /// <param name="text">String to print.</param>
    private void Print(string text) { /* Implementation hidden. */ }
    /// <summary>Print a formatted String to the [Out] Parameter of the Script component.</summary>
    /// <param name="format">String format.</param>
    /// <param name="args">Formatting parameters.</param>
    private void Print(string format, params object[] args) { /* Implementation hidden. */ }
    /// <summary>Print useful information about an object instance to the [Out] Parameter of the Script component. </summary>
    /// <param name="obj">Object instance to parse.</param>
    private void Reflect(object obj) { /* Implementation hidden. */ }
    /// <summary>Print the signatures of all the overloads of a specific method to the [Out] Parameter of the Script component. </summary>
    /// <param name="obj">Object instance to parse.</param>
    private void Reflect(object obj, string method_name) { /* Implementation hidden. */ }
    #endregion
}

```

Figure 24. Default utility functions in C# node

By default, C# script uses System, Rhinoceros and Grasshopper libraries. New libraries can be declared in white field under “using Grasshopper.Kernel.Types” line.

```

using System;
using System.Collections;
using System.Collections.Generic;

using Rhino;
using Rhino.Geometry;

using Grasshopper;
using Grasshopper.Kernel;
using Grasshopper.Kernel.Data;
using Grasshopper.Kernel.Types;

```

Figure 25. Default C# node libraries

```

private void RunScript(int x, object y, ref object A)
{
    One abc = new One(x);
    A = abc.number;
}

// <Custom additional code>
public class One
{
    public int number;
    public One(int input)
    {
        number = input;
    }
}
// </Custom additional code>
}

```

Figure 25. Main and additional methods in C# node

The main method used to interact with input and output is a default “RunScript” method. Initial variables of “RunScript” can be added or deleted only outside the node, as well as changing their types. Variable type can be changed through changing type hint outside the node.

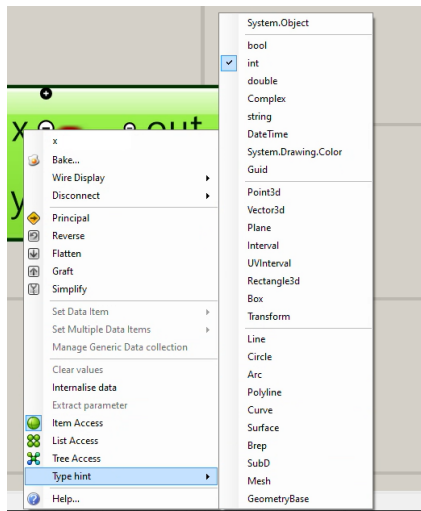


Figure 26. Main method variable type changing.

## **4 Summary**

Nowadays, construction shapes are becoming more and more complex. It is nearly impossible to create that by using a direct modelling approach. That is why parametric modelling specialists will be crucial to have in a big project team in advance.

In this thesis, I collected all information that gives a basic understanding of parametric modelling. Also, the main part of the thesis is a theory about general parametric modelling rules, that can be used not only in Grasshopper but also in other parametric modelling programs, such as Dynamo, Sverchok, OpenCAD, and others. Grasshopper can be a good base to understand general workflow and to use it in real projects.

## References

Jezyk, M. 2019. The Dynamo Primer For Dynamo v2.0. GitBook. Retrieved on 20 October 2022. Available at <https://primer.dynamobim.org/index.html>

Andrew O. P. 2015. The Grasshopper primer (EN) Third Edition v3.3. GitBook. Retrieved on 20 October 2022. Available at <http://grasshopperprimer.com/en/index.html>

Brunelli M. 2017 Parametric vs. Direct modeling: Which Side Are You On? Retrieved on 20 October 2022. Available at <https://www.ptc.com/en/blogs/cad/parametric-vs-direct-modeling-which-side-are-you-on>

Learn Visual Programming. 2020. Grasshopper 101: User Objects | #01 Parameters. YouTube. Retrieved on 20 October 2022. Available at <https://www.youtube.com/watch?v=y7j-czAiaWI>

Learn Visual Programming. 2020. Grasshopper 101: User Objects | #02 Parameters. YouTube. Retrieved on 20 October 2022. Available at <https://www.youtube.com/watch?v=R0JGmKVigEU>

Ayupov A. 2020. Parametric Modelling in Grasshopper – Fundamentals. Stepik. Retrieved on 20 October 2022. Available at <https://stepik.org/course/91457/info>

2022. Parametric model. Wikipedia. Retrieved on 20 October 2022. Available at <https://en.wikipedia.org/wiki/Parametri>

# The Basics of Parametric Modelling

## Course material



**BIM-Integration in Higher and Continuing Education**

Co-funded by  
the European Union



# Course

## Course goal

*Participants:*

- understand the basics concepts of parametric modelling.
- learn how to use Grasshopper and obtain knowledges about further self-learning.
- learn how to connect and use Grasshopper with Tekla Structures.
- the advance method of making libraries in Tekla



**BIM-Integration in Higher and Continuing Education**

Co-funded by  
the European Union



## Preliminary lectures/subjects

- First part is a general information
  - What is parametric modelling and how it can be used?
    - The idea of parametric modelling
    - Node structure
    - Differences between parametric and direct modelling
    - Where PM can be used
    - Main approaches of PM
    - PM software
  - Practical theory
    - Parameters and components
    - Base parameters
    - List and tree
    - Domain
    - Reparametrize
    - UWM Coordinates
    - Rounding
    - Data lancing





## Preliminary lectures/subjects

- **Second part: Tasks**
  - Task 0A: Interface
  - Task 0B: Points and lines
  - Task 1: SINE line
  - Task 2: Vortex arc
  - Task 3A: Attractor
  - Task 3B: Advanced attractor
  - Task 4: Parametric tower
  - Task 5: Parametric staircase
  - Task 6: Connection with Tekla Structures



## The idea of parametric modelling

- Parametric modeling is a type of modelling, concerned with creating geometry based on the mathematical dependencies, pre-programmed rules and algorithms that are called parameters.
- The rules(or algorithms) are usually created in some programs, such as Grasshopper, Dynamo or Blender. The algorithms in those programs are based on a node structure.



- Node modeling is a high programming language. It means that you do not have to write every fundamental algorithmic rule, you can write the algorithm itself by using pre-coded nodes.
- For example: let's imagine the robot, which can not move without orders. In low level(fundamental) programming language we would tell him: "Step right, step forward three times, turn left and etc..", where in node-based language (high level) we would tell: "go to the shop, buy some tomatoes". In other words, the orders in parametric modeling are nodes.

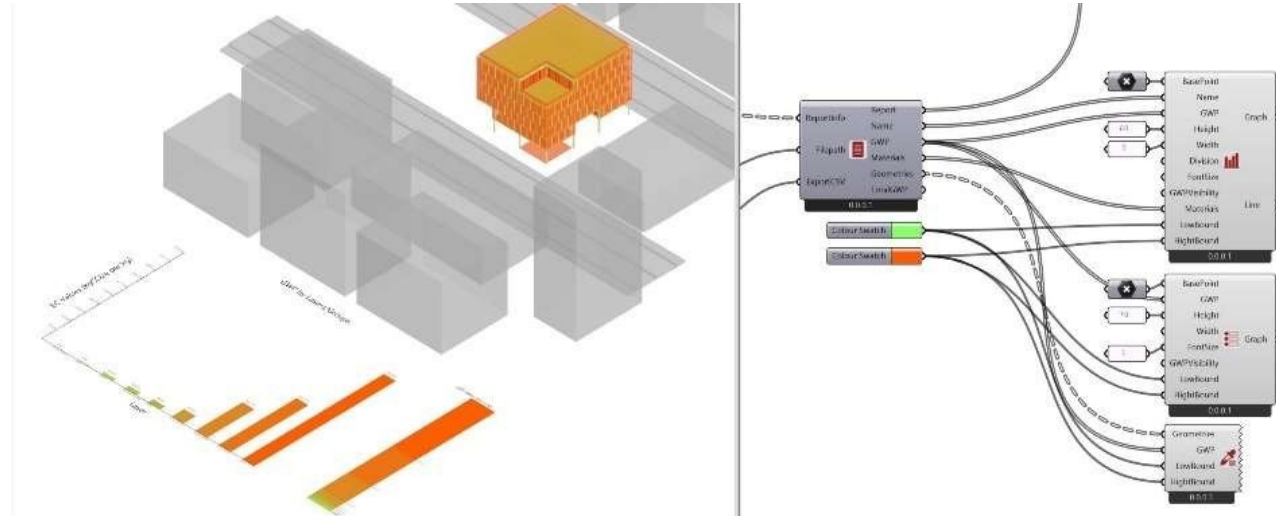


- Cad model is mostly consisting of drawn lines and hatches. The positive side of this modelling approach is a relatively quick result. In the end of modelling process, we have a static model, which means that the cost of one single mistake will be very high.
- Since working on a project without making any mistakes is nearly to impossible, civil engineers nowadays are using the BIM method. The process of creating a project takes longer rather than in CAD, but as a result, we have one solid 3D model that can be easily cut with cross-sections that will be concerned as plans, elevations or joints. That approach means that all mistakes can be easily corrected.
- There is a very strong tool in the field of BIM, called PM. This tool can enlarge the spectrum of actions that we can do with the project.



# Where PM can be used

- With parametric modeling you can create your own plugins, that will help you to manage the project, and they can replace some functions from expensive software.
- You can create and control architecture that has very difficult shapes. And moreover, it will create rules, that can be changed in very little amount of time



# Main approaches of PM

- Generally, algorithms in PM can be divided into two branches: one iteration process (single result) and multi-iteration processes (generative design).
- One iteration process is an algorithm that creates one result with one set of initial data, while in the generative design approach one piece of initial data can be transformed into a variety of results, or a list of best results, according to following rules.



- Nowadays there is a large range of PM software. The most popular programs are Grasshopper, Dynamo for Revit and Blender. The Blender is mostly used for games and cartoon animations. The Dynamo is used for making plugins in Autodesk Revit. The most popular for civil engineering now is the Grasshopper. You can use Grasshopper for a wide range of BIM programs, such as Tekla structures, Revit and Rhino.
- In this course Grasshopper will be looked at, since it is usable for Tekla.
- Grasshopper originally was a plugin for Rhinoceros 3D. As other PM software it uses node system of creating algorithms. We will talk about basic programming concepts and methods. All theory and basics will be shown in connection with Rhino and the last part of the report will be related to the connection between Grasshopper and Tekla.



# Practical theory



*BIM-ICE – BIM-Integration in Higher and Continuing Education (KS1905)*

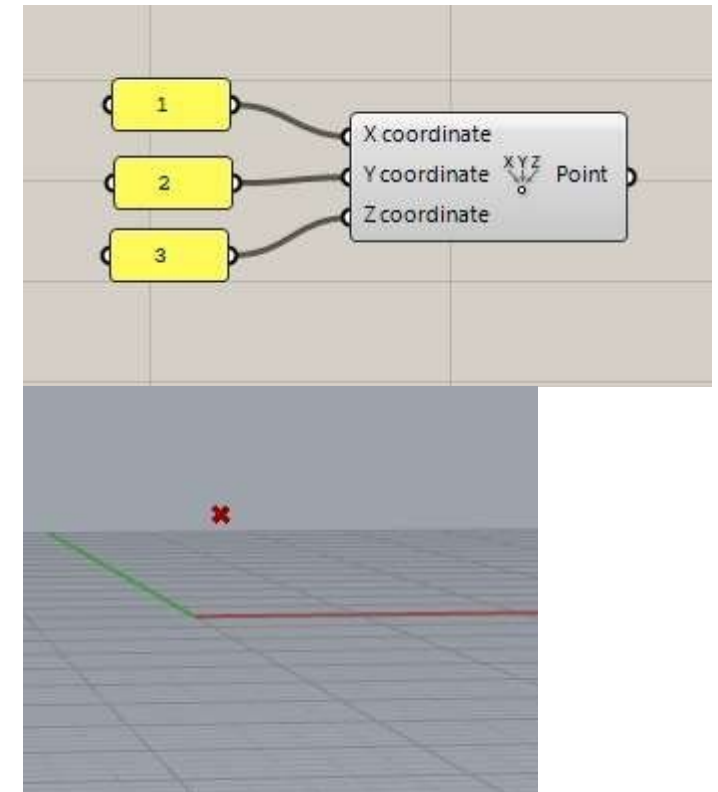
*Funded by the EU, Russia and Finland via the South-East Finland – Russia CBC 2014-2020 programme*





# Parameters and components

- There are two types of nodes in Grasshopper: nodes of parameters and nodes of components.
- Nodes of parameters are used as an input. Input data can be made in Grasshopper, taken from Rhino, Tekla or etc., it can also be taken from other files, such as Excel or .txt files. It can be numbers, points, lines colors and etc..
- Component nodes are used to transform parameters into other objects. For example, we have three parameters: “1”, “2” and “3”. We can plug them into a component node called “Construct point”. This node requires three numbers to input, and the output will be one point. So, the question is, is this point a parameter or component? In this situation it should be called a component, because the point can only be moved by changing the initial data (numbers from the input).



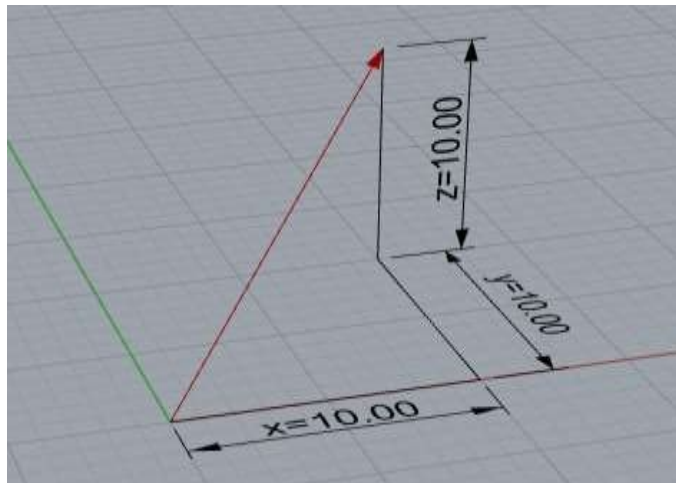
- Number (or float, or real) is every number from  $-\infty$  to  $+\infty$ , including numbers with coma. Talking about root numbers or irrational numbers, usually a computer makes them real by rounding it. (2,5; 59,38; -500,69)
- Integer is every number you can count with, including 0 and negative numbers (5; 89; -53)
- Text is a set of symbols. This parameter may consist of any letters of any alphabet, symbols or numbers. In Grasshopper text and number can be recognized by special parameter nodes



- *Boolean* is a logical parameter that can be “true” or “false”. Usually, Booleans are used to switch nodes working mode, or to sort items in a list.
- *Point* is the simplest geometry. It can be used as a coordinate system origin or the reference for geometry. For example, a line can be built between two lines, a surface can be built with three reference points. Usually, a point can be defined by three numbers. These numbers are coordinates. A point can be built in cartesian(x,y,z), uvw, cylindrical or other coordinates, but despite the method of creating, a resultant point will be defined with x,y,z coordinates.



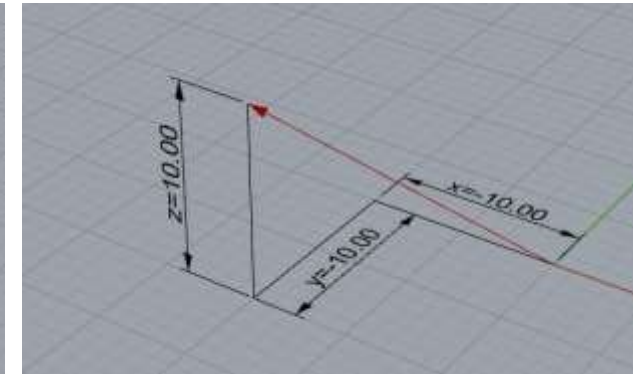
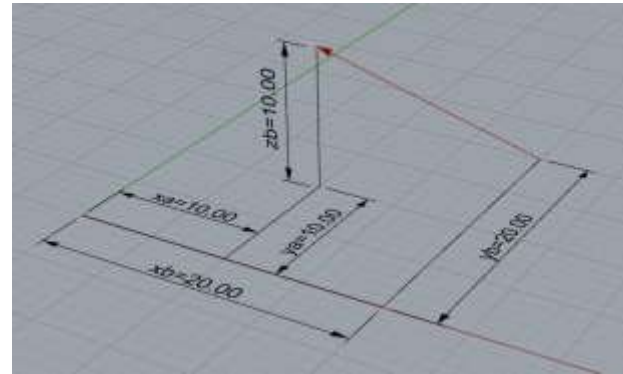
- Vectors are the abstract geometry that helps to define the location, orientation and spatial context for other geometry. A vector is a geometric quantity describing direction and magnitude. A vector can be defined by three numbers, and it will be written like a point. To distinguish a vector from a point let's draw three pictures.



This is a vector, which can be described in written form as “{10,10,10}”. It means that the starting point(origin) that we used to build this vector has coordinates “{0,0,0}” and the second(vector end) “{10,10,10}”.

- Now let's investigate next two examples

- These two vectors have the same written form “{-10,-10,10}”. Let’s find out why. The points of



the first vector “{20,20,0}” (start) and “{10,10,10}” (end). Second vector:

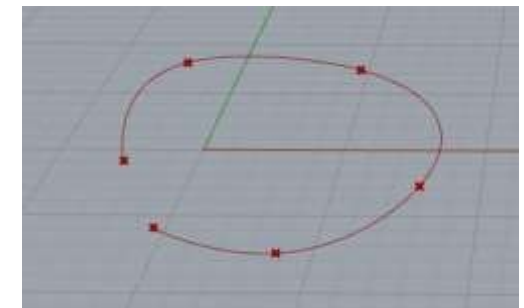
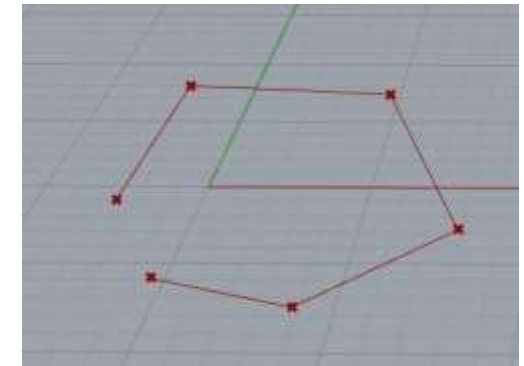
“{0,0,0}” (start), “{-10,-10,10}” (end). Let’s define first vector:

$$\{10-20, 10-20, 10-0\} = \{-10, -10, 10\}$$

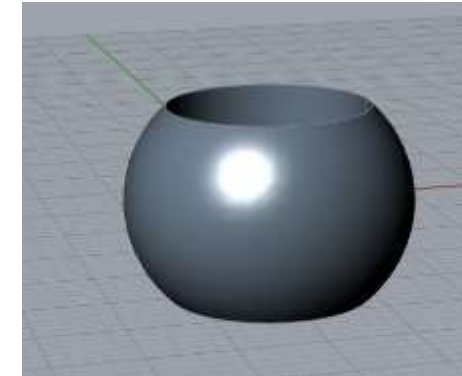
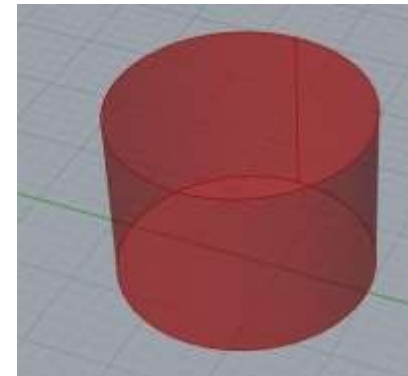
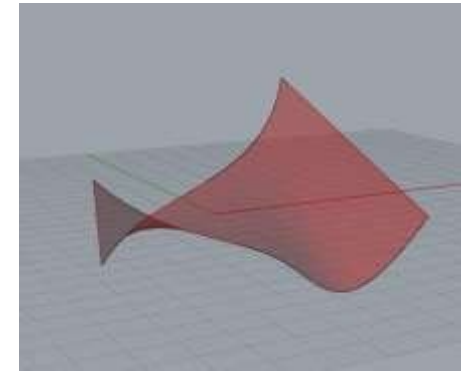
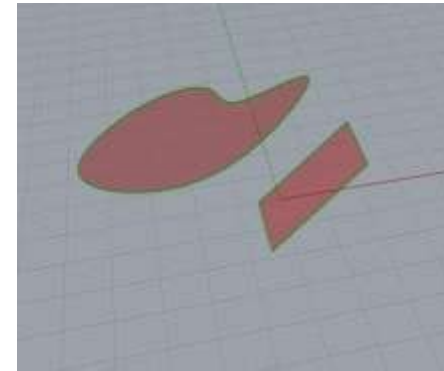
and the second vector:  $\{-10-0, -10-0, 10-0\} = \{-10, -10, 10\}$

So, it means that the vector doesn’t have a fixed location. Vector shows only direction and its length

- Line is an object that usually connects two points (or interpolates through best fit collection of points). It can be defined by a start point and an end point.
- Polyline is a curve that connects more than two points by straight segments. It consists of lines and vertices. These types of curves can be closed.
- Spline is a smooth curve which interpolates through the vertices, or control points. Spline, like a polyline can be closed.



- Surface can be explained like a “Infinitely thin shell”. Surfaces can either lay on a one plan or have multiple plans in different points.
- Brep is every 3D solid object. The surface is a special case of brep.



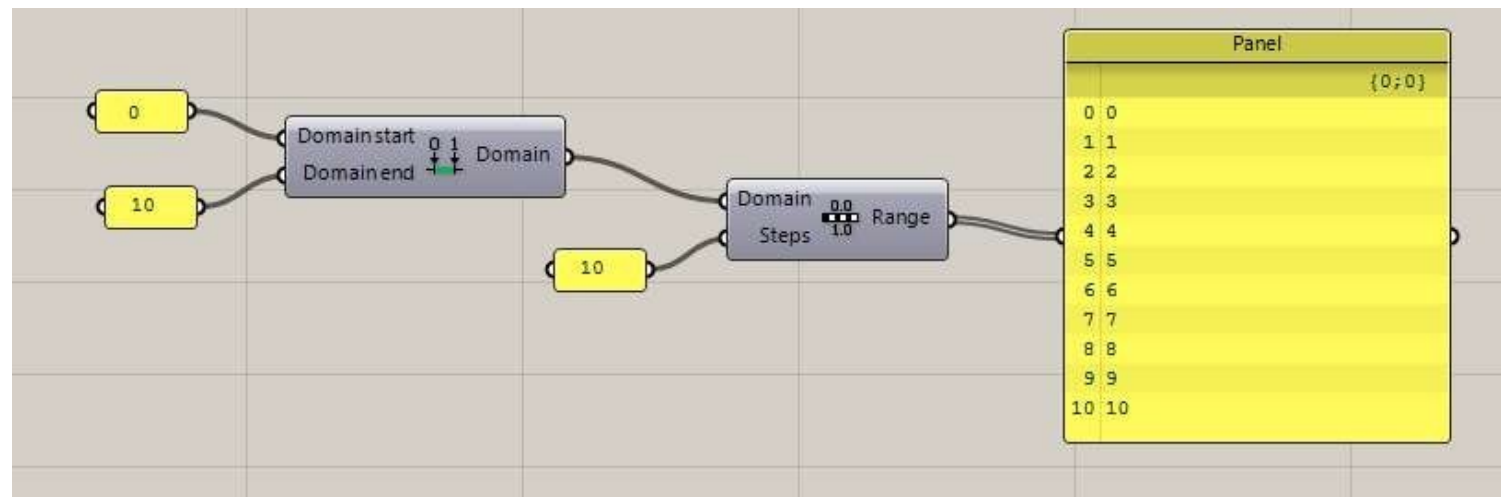


- A list is a structured collection of data. Each list has lines with data. Each line has a unique index. The first line of the list has “0” index, the next line will be with index “1”, then “2” and etc... Lists can be used to collect any data.
- Data tree (list of lists). To understand what the data tree is, let’s imagine a list of tasks. If we want to find the task for today, we need to find the third line in the list. This page is part of the notebook. Now, if we want to find the task for today, we need find “page 13, line 3”
- Now let’s imagine a shelf of notebooks. If we want to find the task, the path will look like “notebook “Daily tasks”, page 13, line 3”, and so on.



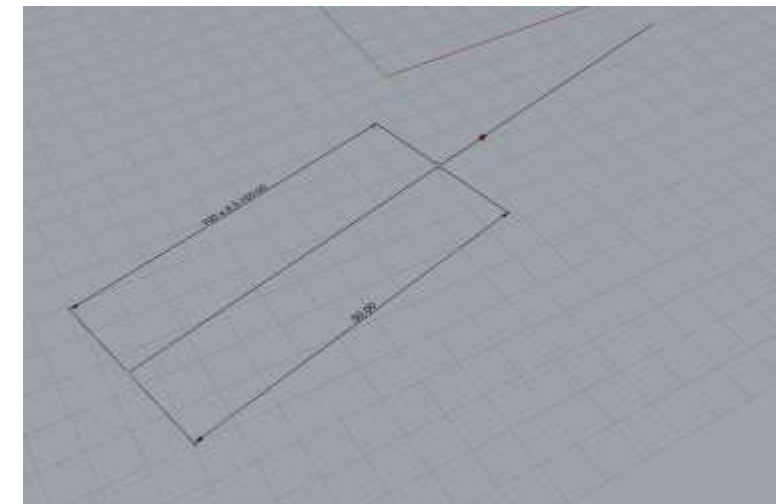
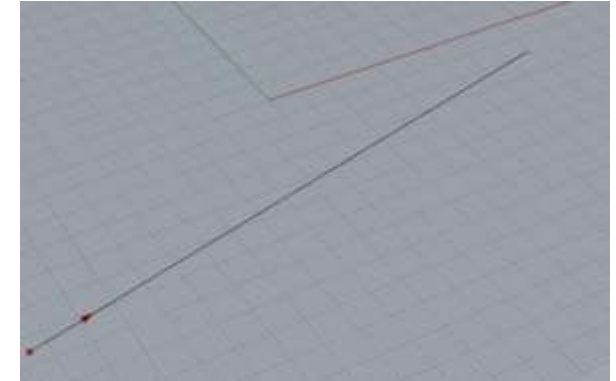


- Domain is an infinite number of numbers in a chosen interval. Usually, domains are used to create a range of numbers. For example, we can create a domain from 0 to 10 and divide it into 10 steps. The result will be a list with numbers

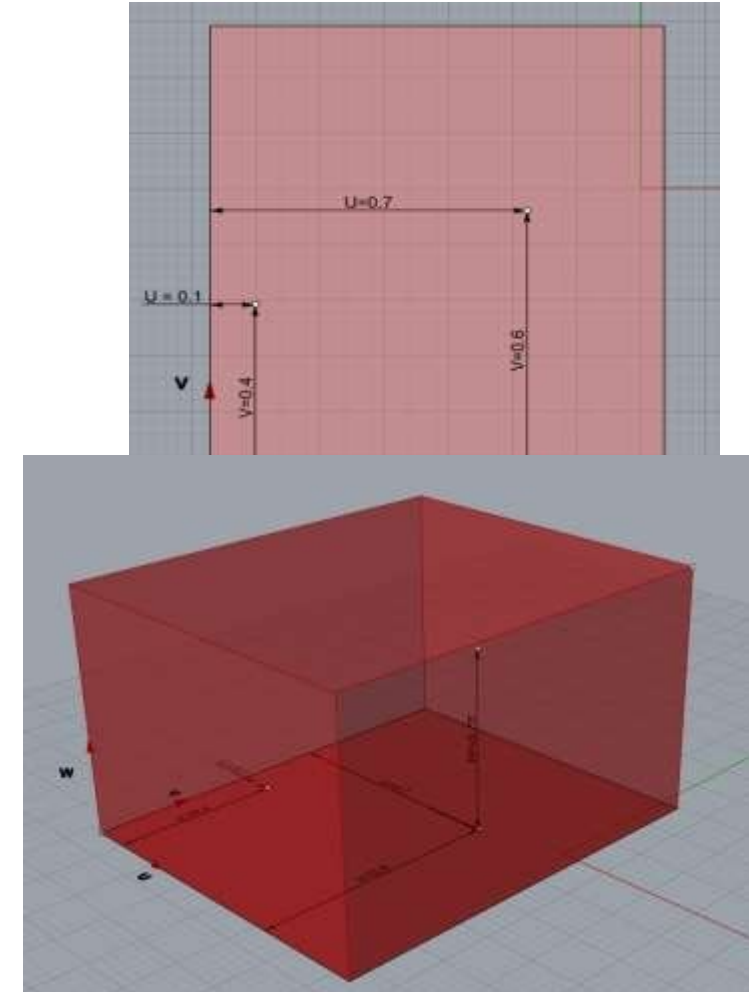


## Reparametrize

- Reparametrize is the tool, that converts geometry dimensions into domain from 0 to 1. For example, a line is drawn in the picture below.
- Now let's create the point on it. As a coordinate system we will use 1D coordinates, that goes along the line. The length of this line is 100 mm. So, in order to put this point in the middle of the line, we need to use coordinate 50mm.
- Or we can evaluate the length of this curve and multiply it by 0.5. Then, we can do the same with all points along the line ( $35\text{mm} = 100\text{mm} * 0.35$ ,  $66\text{mm} = 100\text{mm} * 0.66$  and so on...). That what the tool reparametrize do. This concept is usually used for evaluating points, normal and plans in curves, surfaces and breps.



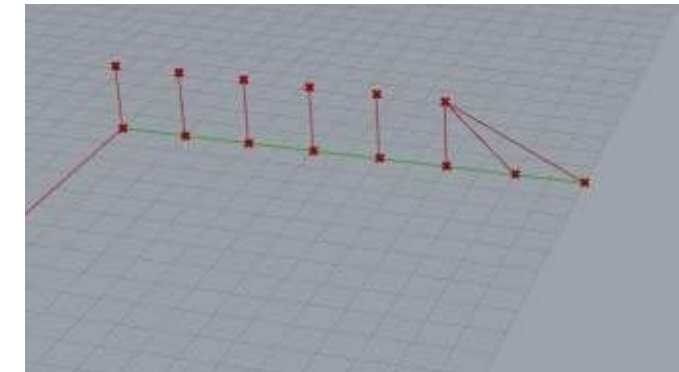
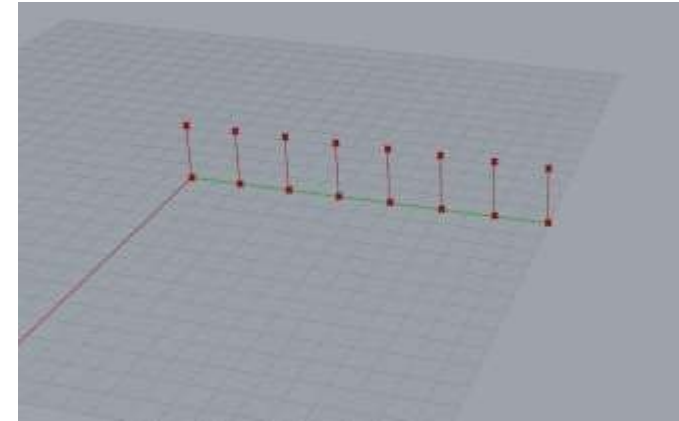
- U,V,W coordinates are the numbers that are used to define point position on a surface. Their values are usually between 0 and 1.
- U and V coordinates are usually used when we talking about surface, but in solid objects W coordinate is also used.



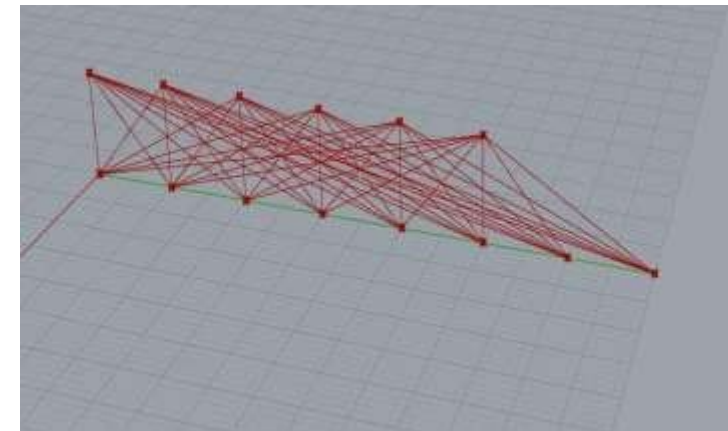
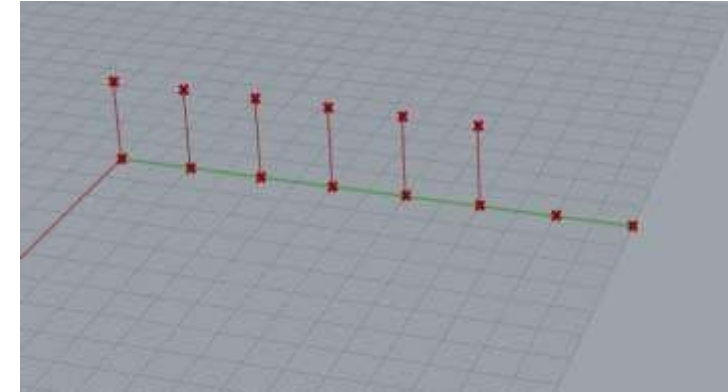
- There are three types of numbers rounding in Grasshopper:
- Floor - rounds to the closest smaller integer
- Ceiling - rounds to the closest bigger integer
- Nearest - rounds by the basic math rule (0.49 goes to 0, 0.5 goes to 1)



- The lancing is the rule for interaction between two lists.
- Let's create two sets of points and connect them with the lines.
- Now let's change the number of points in one list. As you can see, the last point of the upper set of points connects with the three last points of the second list. It is called "Longest lancing". It is a standard working mode for all Grasshopper nodes.



- Now, let's change the lancing. Here you can see that only points with the same indexes are connected. This called "Shortest lancing"
- And finally, let's add the node that connects all points in all orders. This lancing method called "Cross reference".
- All this lancing is used not only to connect points, but also for interactions between two data sets.



Task 0A

# Getting started with the interface



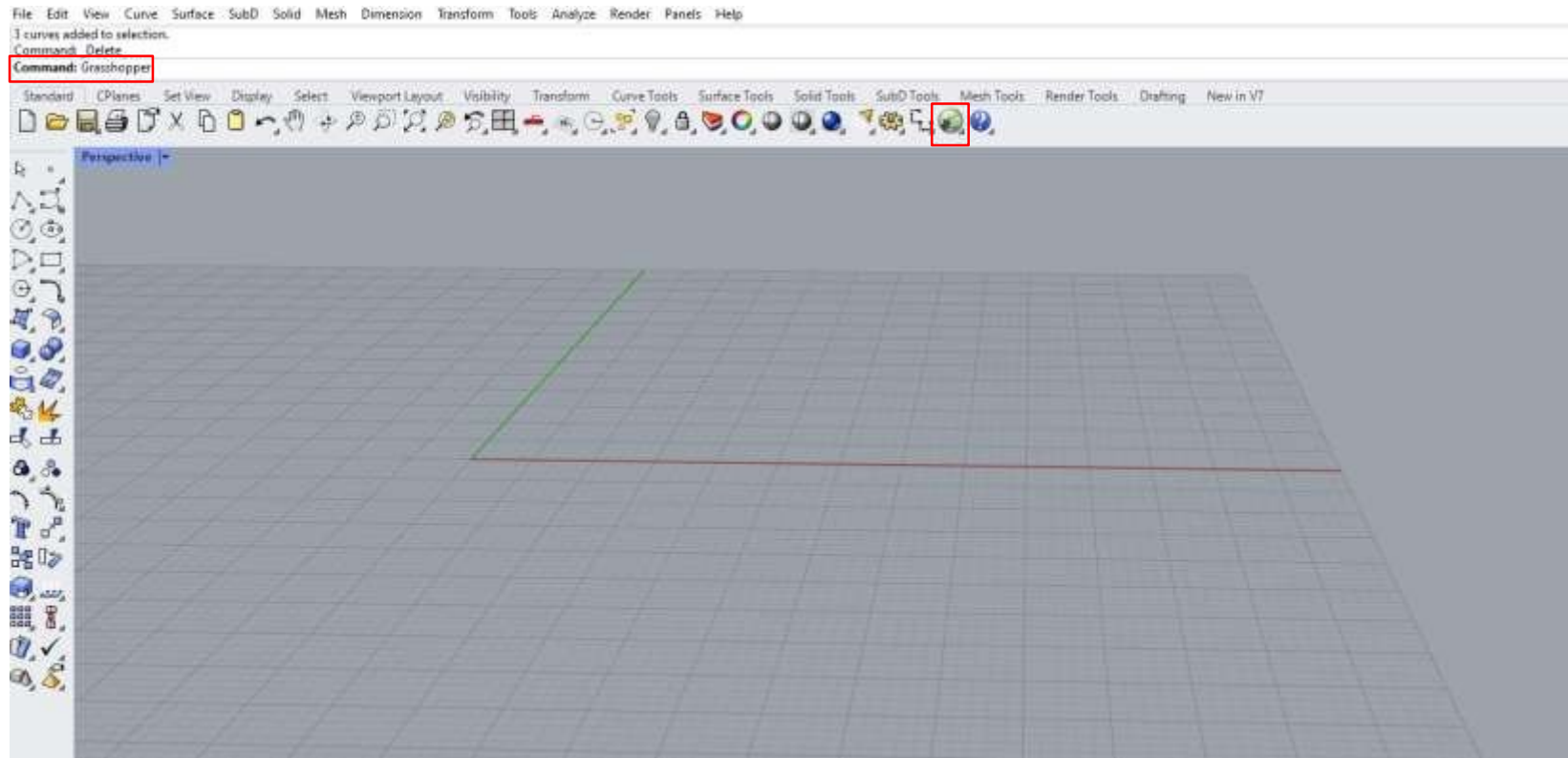
*BIM-ICE – BIM-Integration in Higher and Continuing Education (KS1905)*

*Funded by the EU, Russia and Finland via the South-East Finland – Russia CBC 2014-2020 programme*



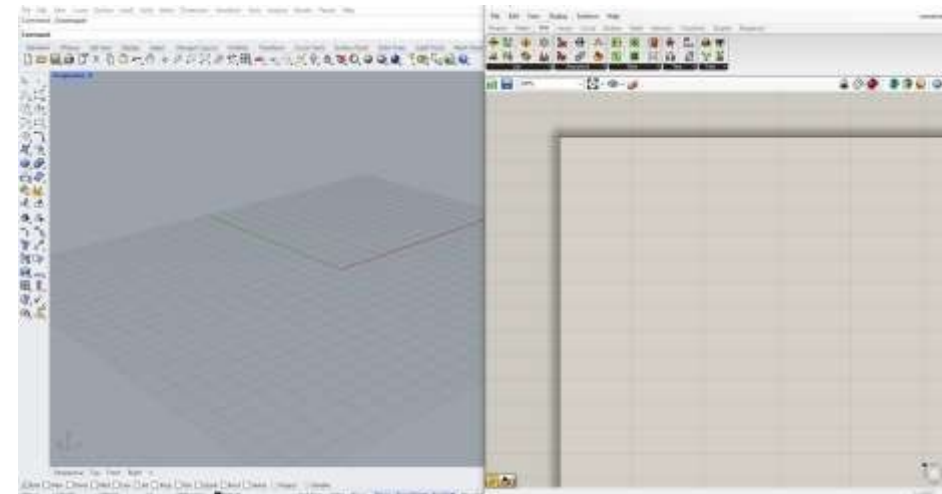
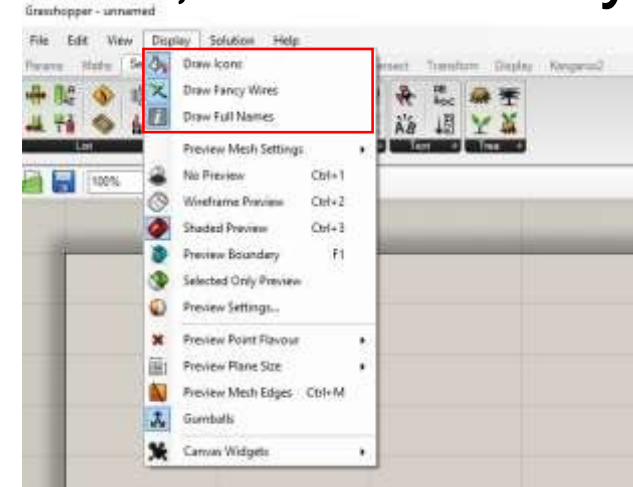


- In order to open Grasshopper, first, you need to open Rhinoceros. In this course we will use Rhinoceros 7.
- In Rhino you can type “Grasshopper” in command line (1), or press GH icon (2)

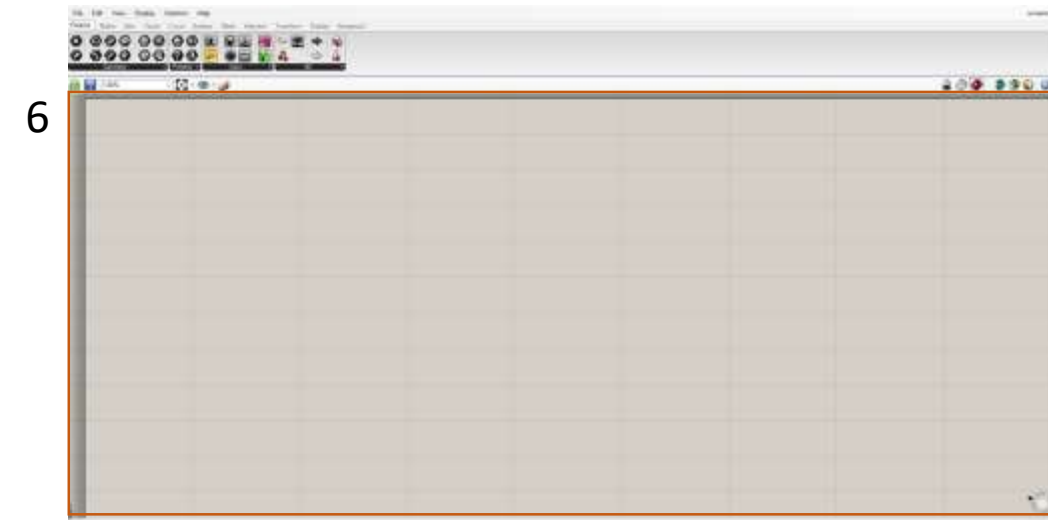




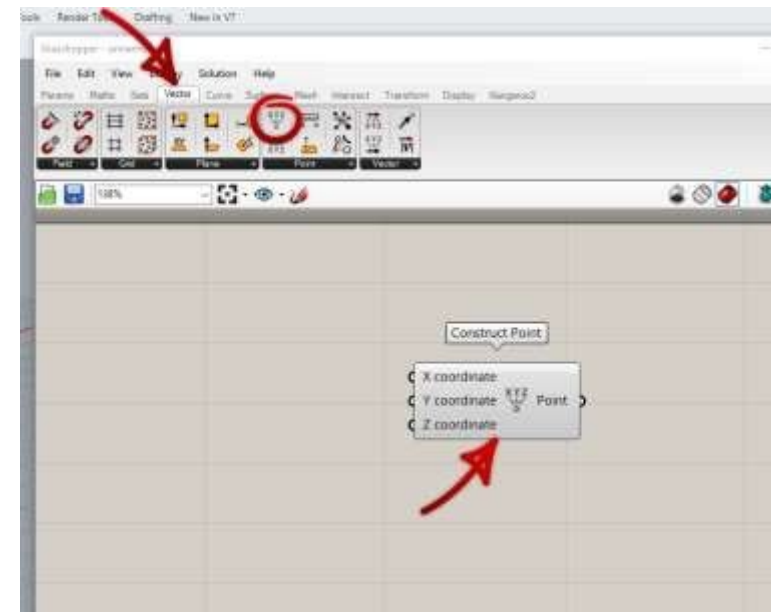
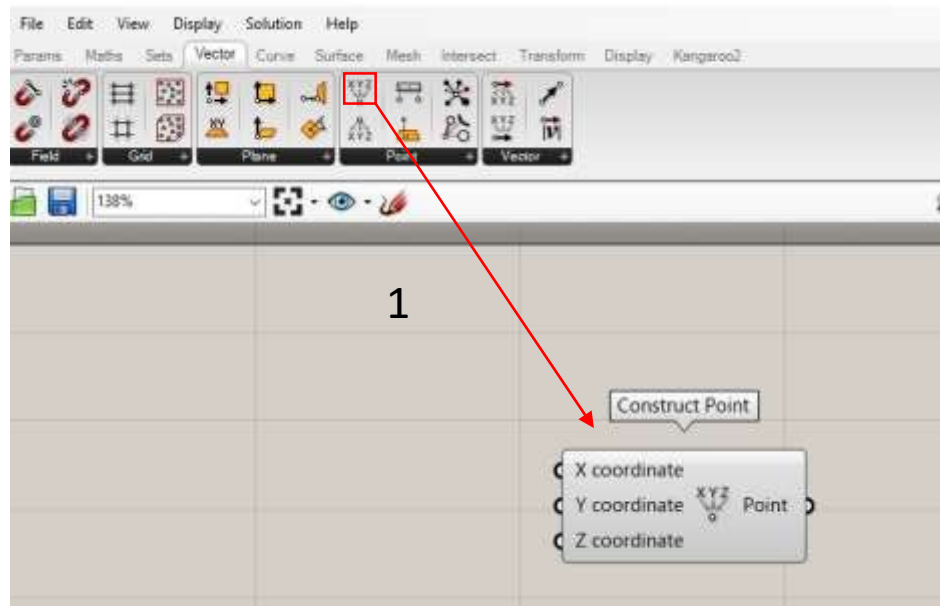
- Press File-New document
- Press Display and make sure, that “Draw Icons”, “Draw Fancy Wires” and “Draw full names” are enabled
- One of the most comfortable way to use Grasshopper with Rhino, is to make Rhino in full screen and drag GH to the right side of the screen (if you do not have second monitor)



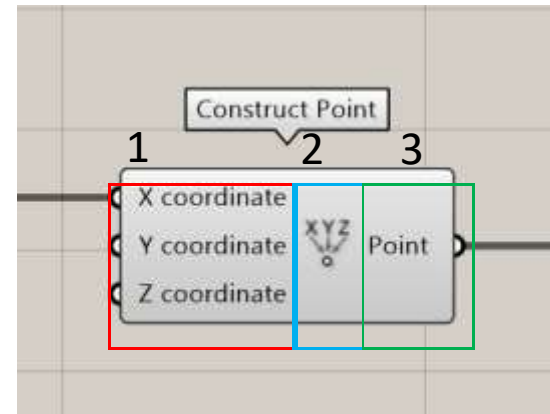
- GH main interface can be divided into settings (1), node pages (2), node panels (3) and nodes (4).
- In usual panel working mode you only see the part of all nodes. To view the rest nodes, you can press (5).
- To apply quick search you can do a double click to a canvas (6). The keywords that you need for search will be shown in the examples (7).



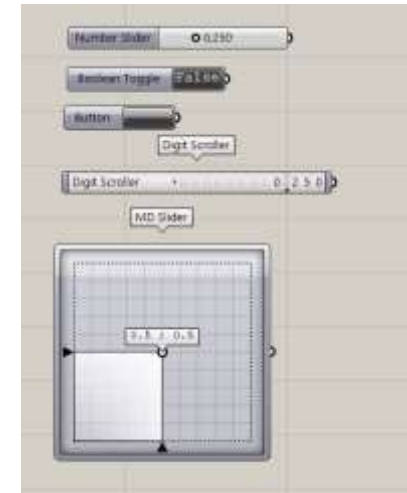
- Now, we will talk about nodes usage. In order to put node to canvas, you can drag the icon of node from panel to canvas, or to click the icon than click on canvas (1).
- Also, by pressing CTRL+ALT+LMB on the node in canvas you can inspect the node location (2). This tool will show you where to find this node.



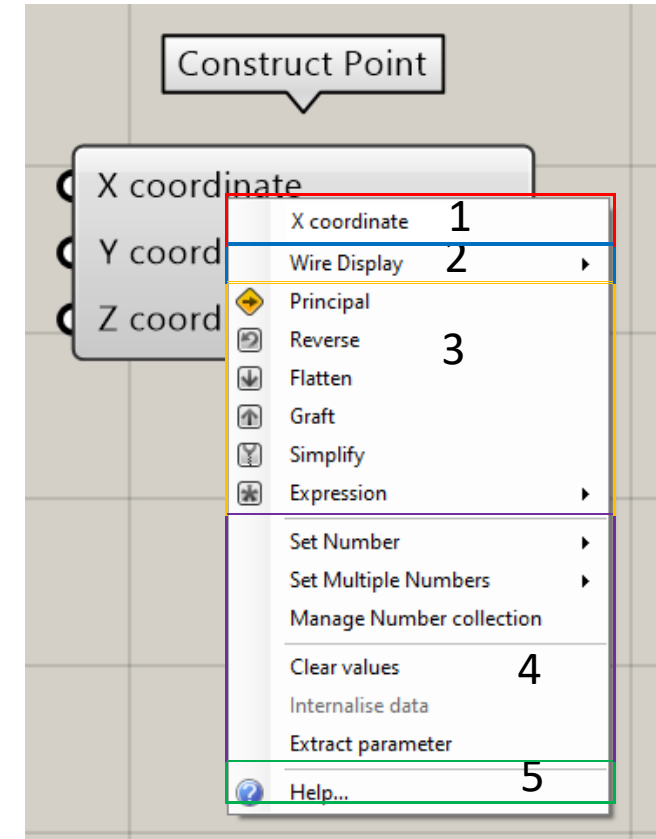
- The node have input slots (1), icon (2) and output slots (3). Input slots are always on the left side of the node, icons in the middle and outputs on the right side.
- Some nodes have no input slots. Those nodes are called as input nodes (4). They are used to generate numbers, Booleans, text and etc... Inside the GH.



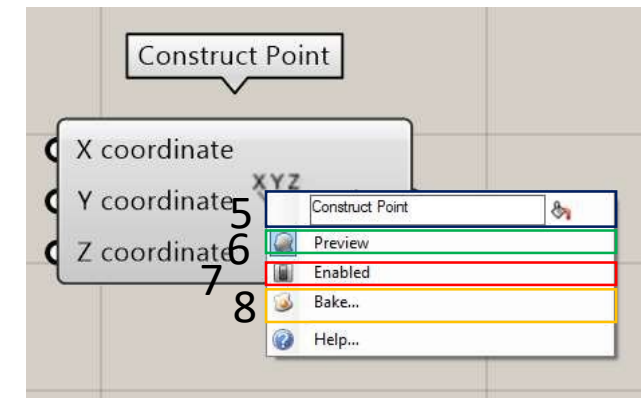
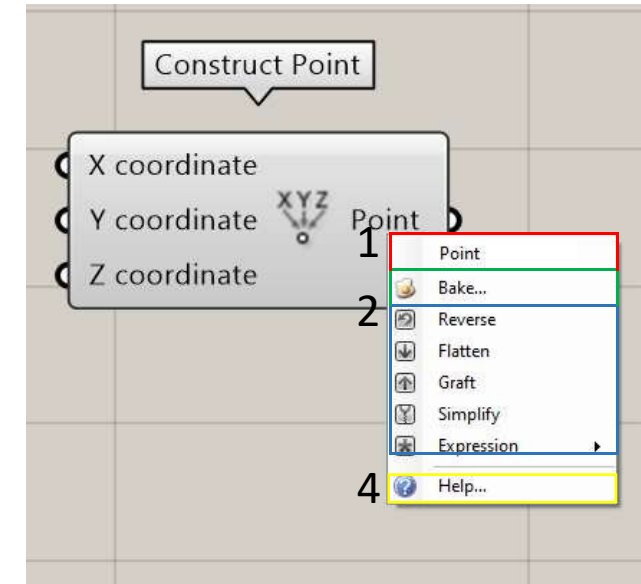
4



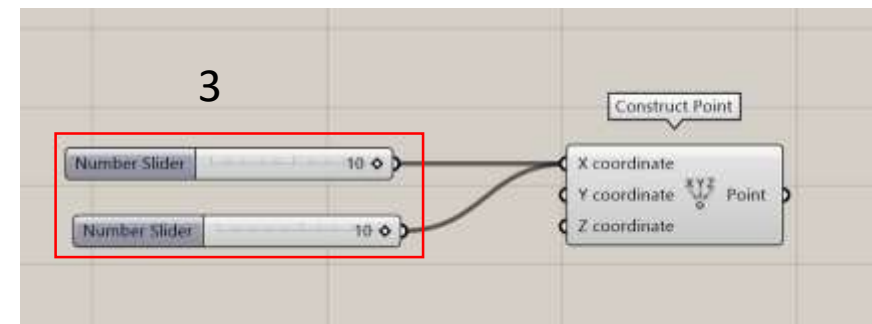
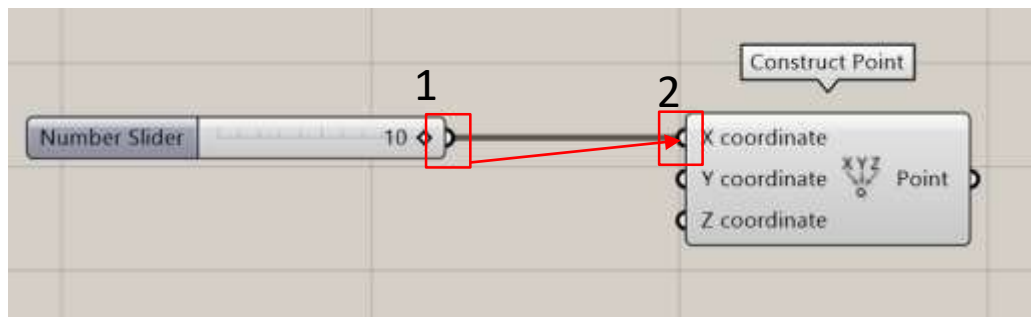
- Right clicking on the input slot will cause the menu, that can be sued for:
  - Changing the name of the slot (1)
  - Changing the wire display (2)
  - Applying additional operations inside the node (3)
  - Setting default values and managing input data (4).
  - Instruction (5).



- By right clicking on the output slot you can:
  - Change the name of the slot (1)
  - Convert GH geometry into Rhino geometry (2)
  - Add operations (3)
  - And see the instruction (4)
  
- By right clicking on the icon you can:
  - Change the node name (5)
  - Show or hide the preview of the node (6)
  - Enable or disable the node (7)
  - Convert GH geometry into Rhino geometry (8)

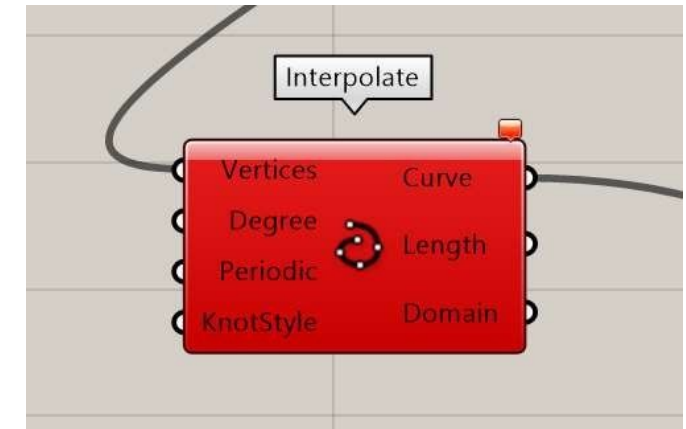
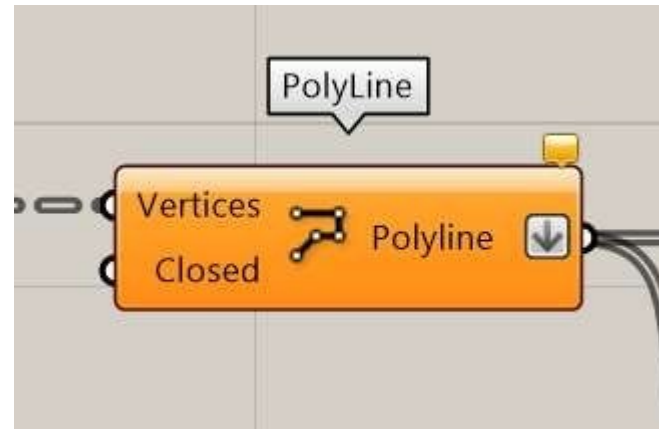
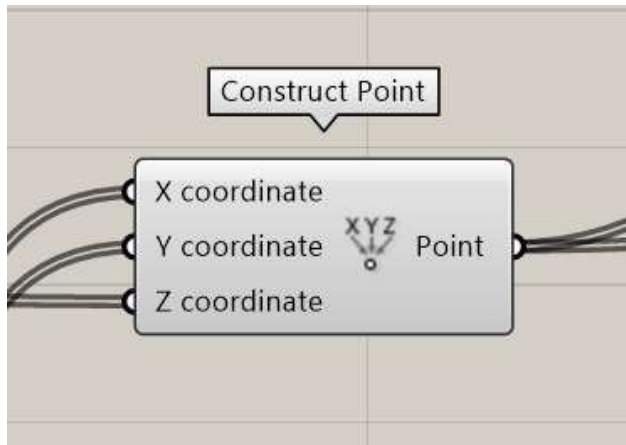


- To connect nodes, drag the wire from output slot of the first node (1) to the input slot of the second (2).
- There are two ways to disconnect nodes:
  - By right clicking the input slot and pressing disconnect button
  - By dragging wire from slot (2) to (1) while holding CTRL button
- There is possibility to connect multiple outputs into one slot (3). To do it you need to hold the SHIFT button, while connecting second or other wires.





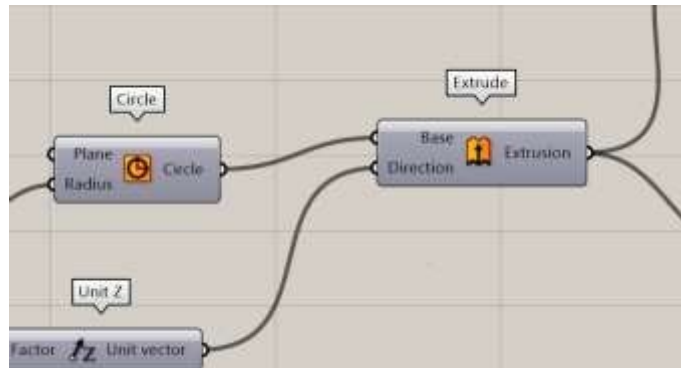
- Nodes can be coloured into four colours: white, grey, orange and red.
- White colour means that node works properly.
- Grey colour same as white, but preview is disabled
- Orange means that node works with mistakes (but still works).
- Red means that node cannot work because of wrong input data



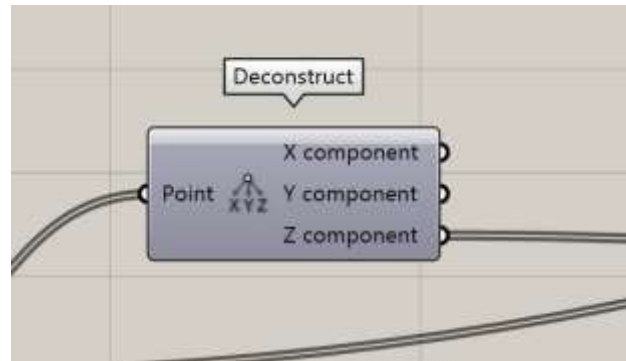


- Different types of wires means different types of data that they transfer:
  - Thin wire mean that it transfers one object (single data) (1)
  - Thick wire mean that it transfers list of data (2)
  - Wrecked wire mean that it transfers data tree (3)

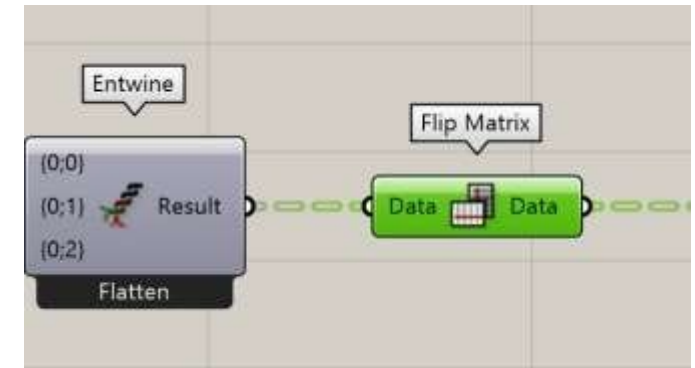
1



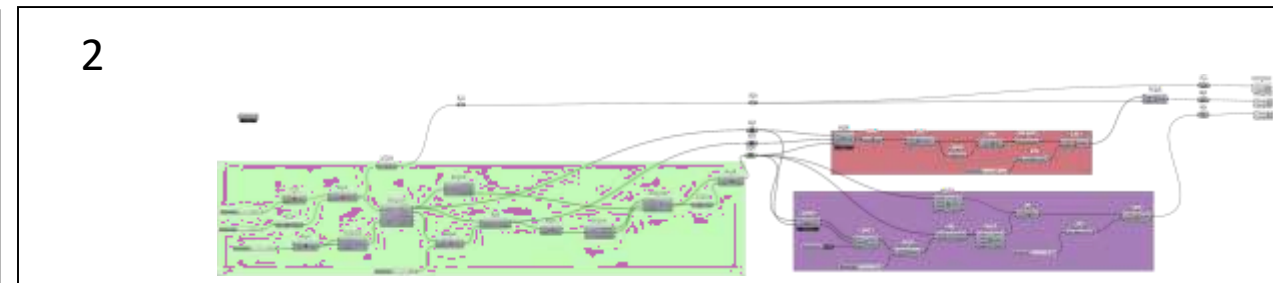
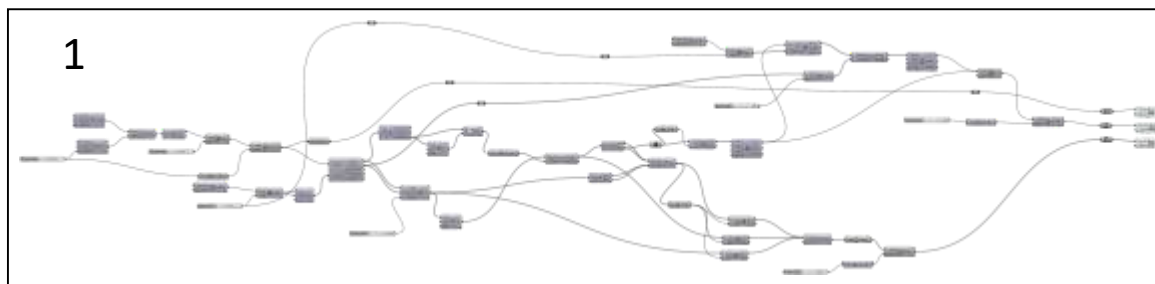
2



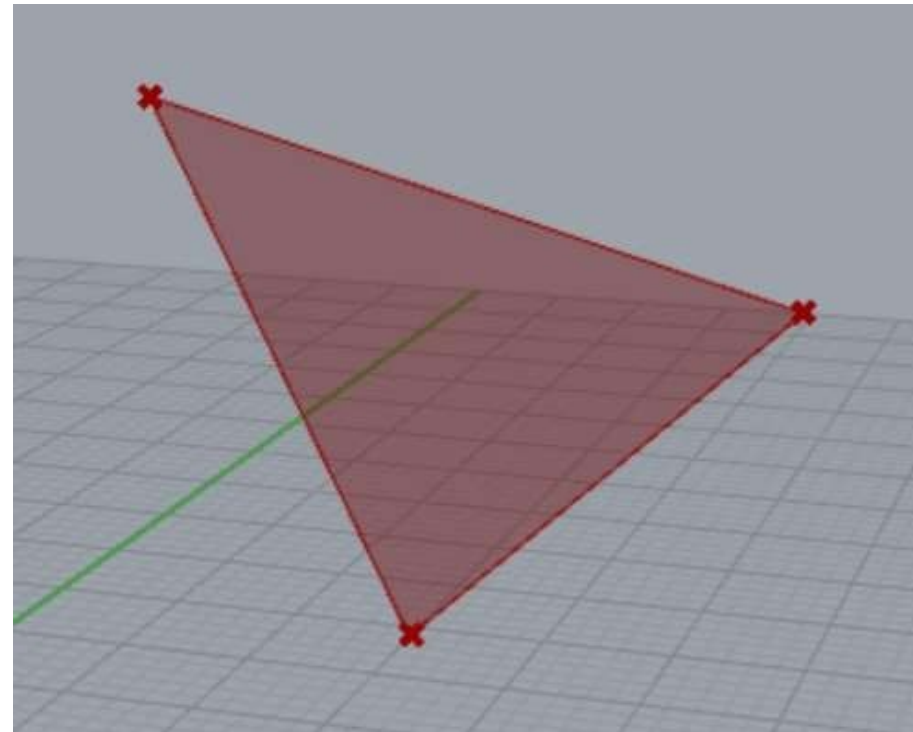
3



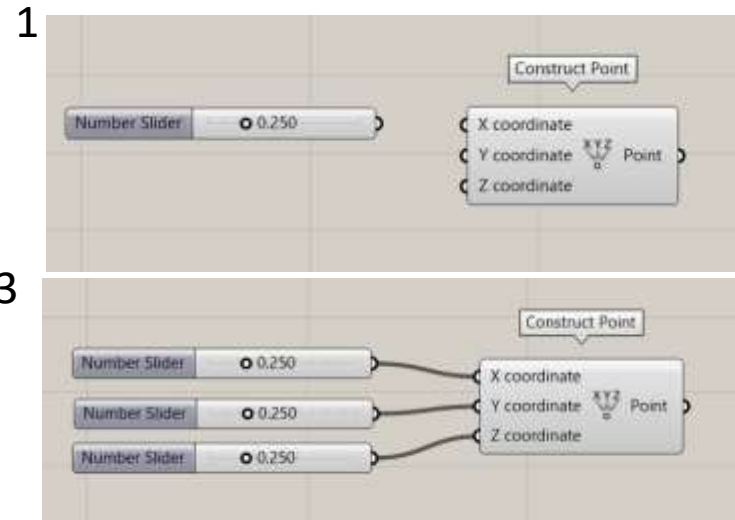
- To copy nodes you can choose them, press CTRL+C and then press CTRL+V, or you can choose required nodes, drag them and WHILE dragging press and hold ALT.
- In order to avoid mix-up in big algorithms it is better to group nodes into patterns. Sometimes it totally impossible to change or to fix old algorithm just because it was not grouped properly. In picture (1) you can bad example and in picture (2) – good one. To group nodes you need to choose nodes, right click to the canvas and choose group.



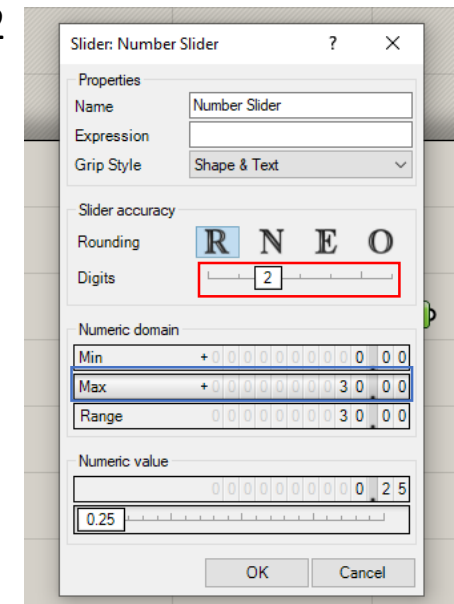
## Task 0B Points and lines



- A) Open Grasshopper
- B) Press File - New document
- C) Let's start with node Construct Point. You can find it by going to Vector-Point-Construct Point, or by calling quick search (double click on canvas) and typing "Construct point".
- D) Call the node Number Slider (1).
- E) Make a right click on Number slider and choose Edit... Change the number of digits to 2 and max number to 30 and press OK (2).
- F) Copy Number slider 2 times and connect first to the X coordinate slot, second to the Y coordinate slot and third to the Z coordinate slot (3).
- G) Drag sliders and look how the point in Rhino behave.

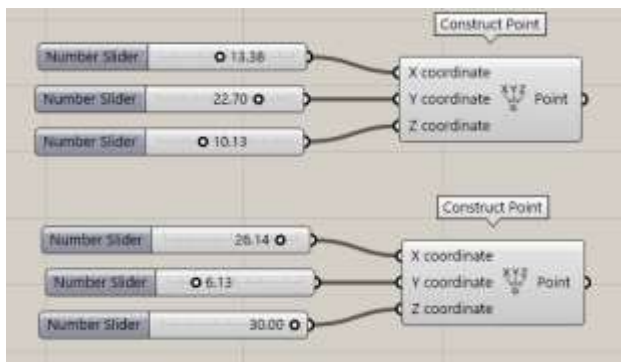


2

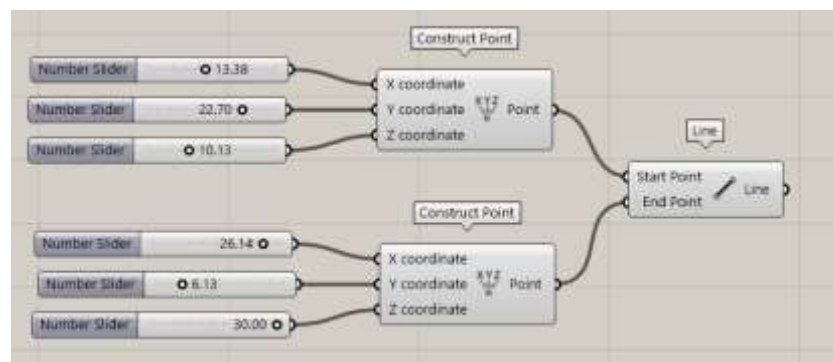


- H) Copy all nodes and change values on sliders (1).
- I) Add node Line and connect it with points like shown on a picture (2)
- J) We can make a triangle by adding one more point and two more lines like in a picture (3)

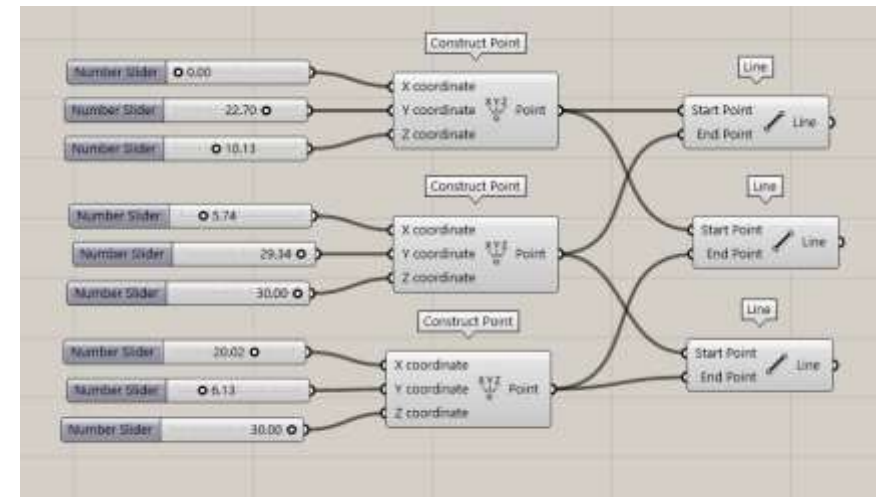
1



2

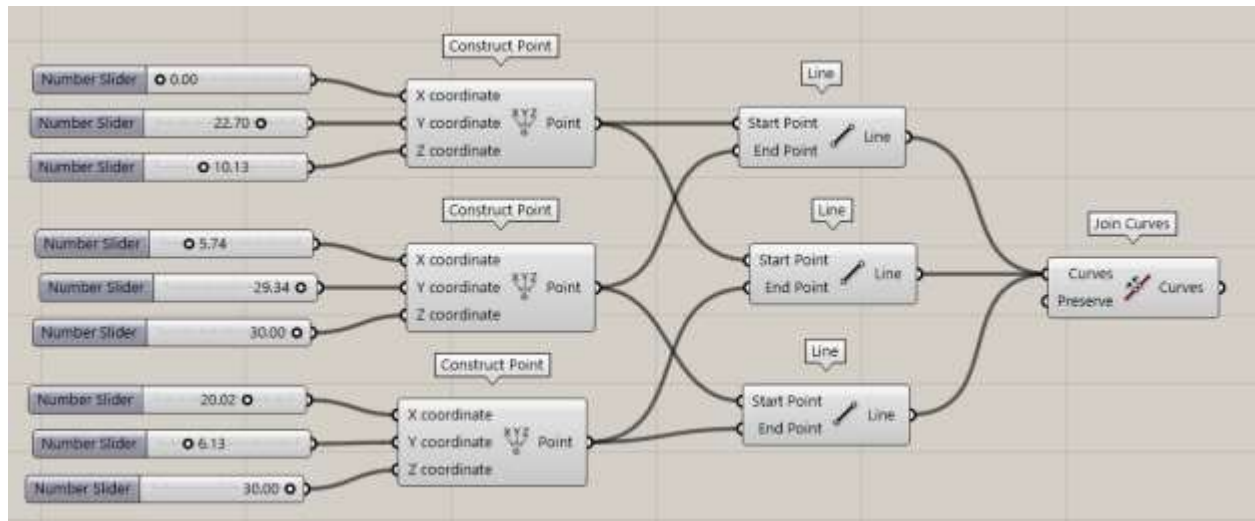


3

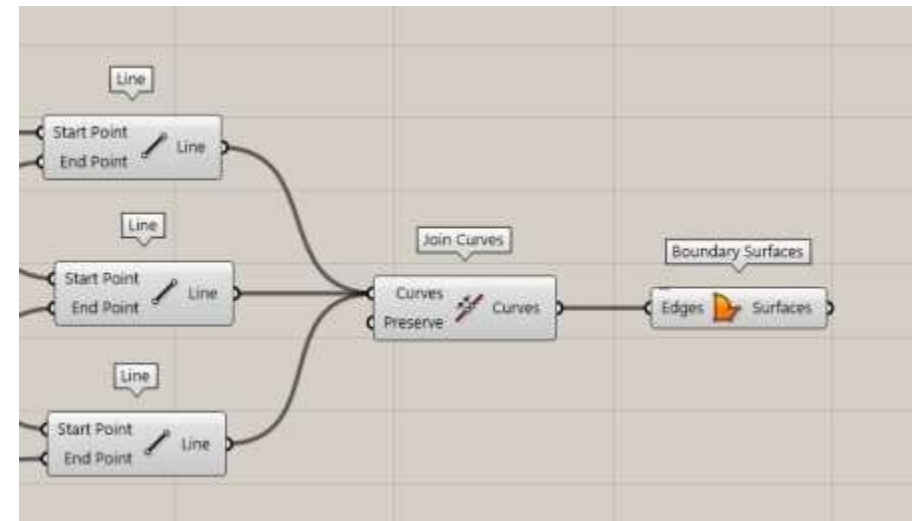


- K) Add node Join Curves. Plug all three lines into Curves slot. To do so, make sure that you are holding Shift button while connecting (1).
- L) Those three joined curves will make one closed polyline. With that polyline we can make a surface by plugging them into Boundary surface node (2).

1

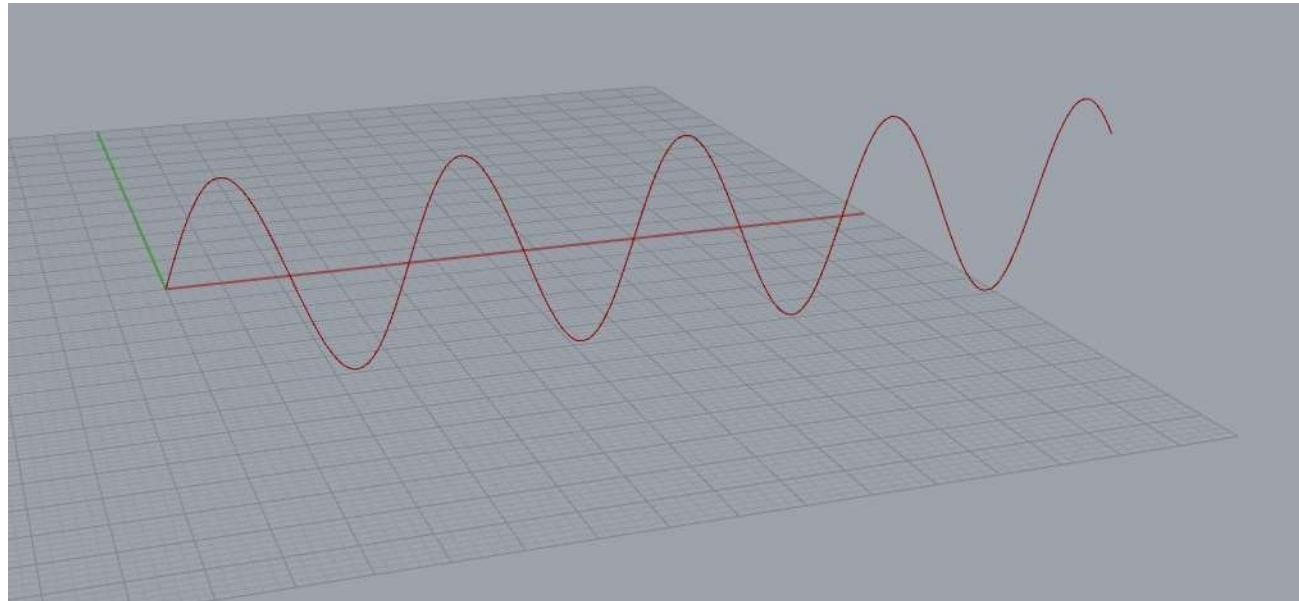


2

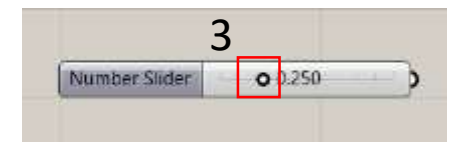
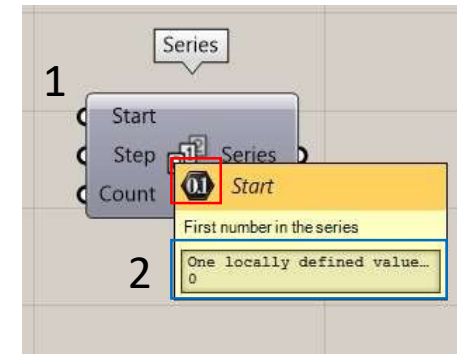




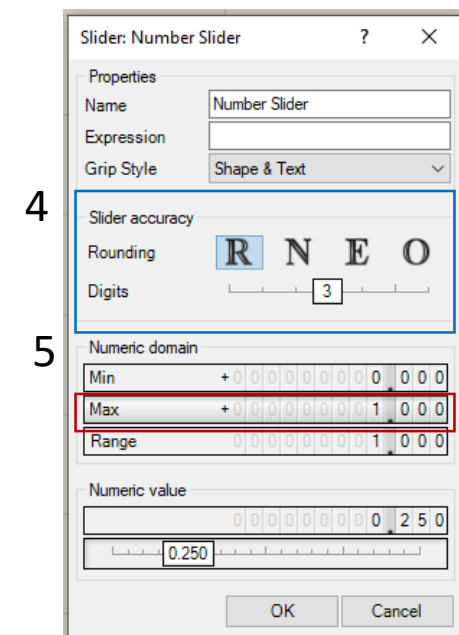
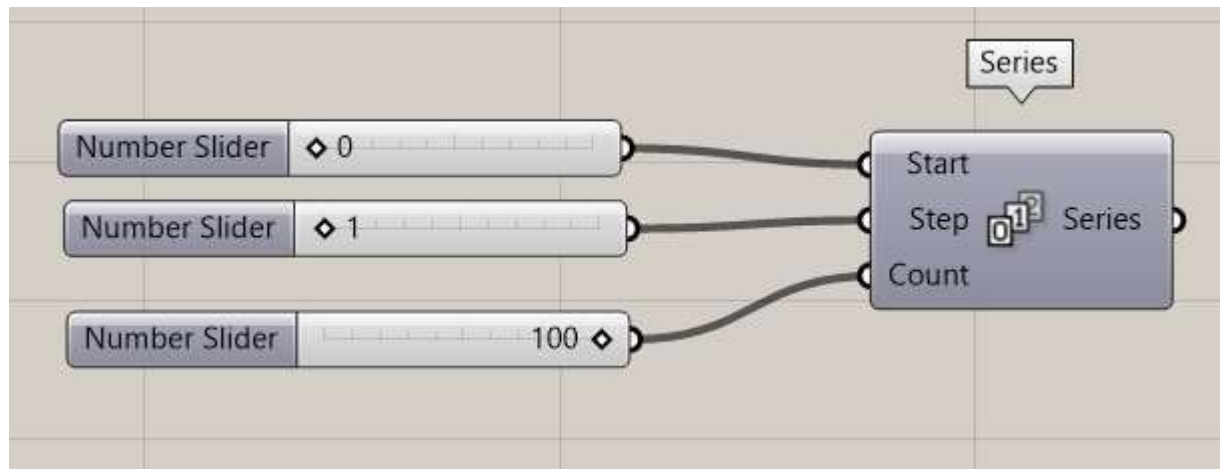
## Task 1 SINE line



- A) Find the node “series”. This node creates the series of numbers with certain step. To find out the data that it require we can point cursor to the input slot. The icon (1) shows us that we need to use the number. The hint (2) tells that this slot have default number. This is the easiest way to understand how the node works.
- B) Call the node “Number slider”. We can slide this toddler (3) and change the number.
- C) In order to change the limits and number type of the slider, right click to the node and press “Edit”. Change rounding from float to integer. Then double click to “max” (5) and change to 100.
- D) Copy this node three times and plug them into series input slots (6)



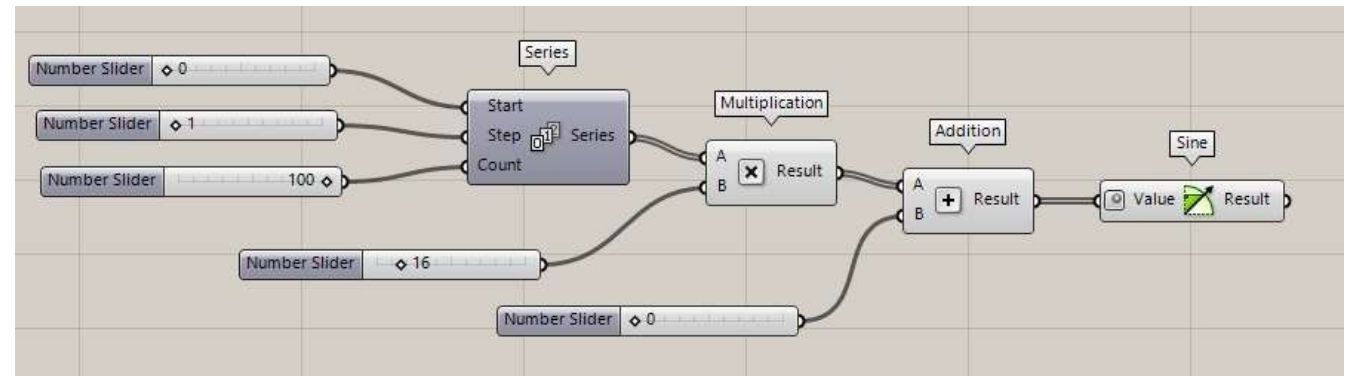
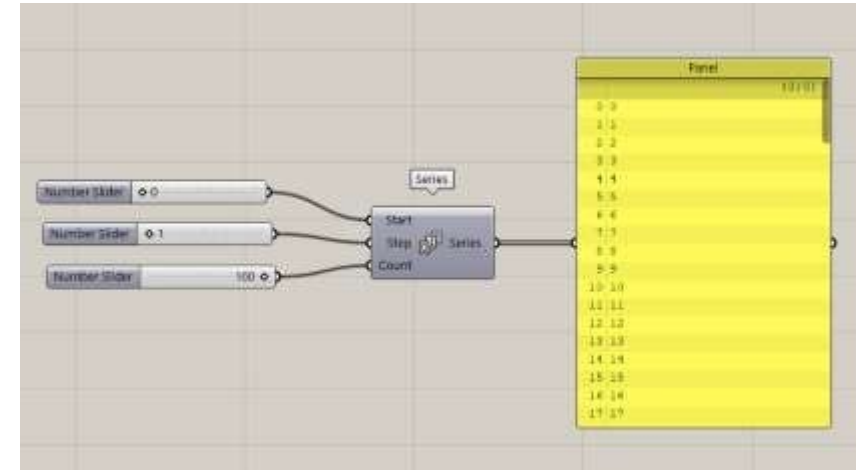
6



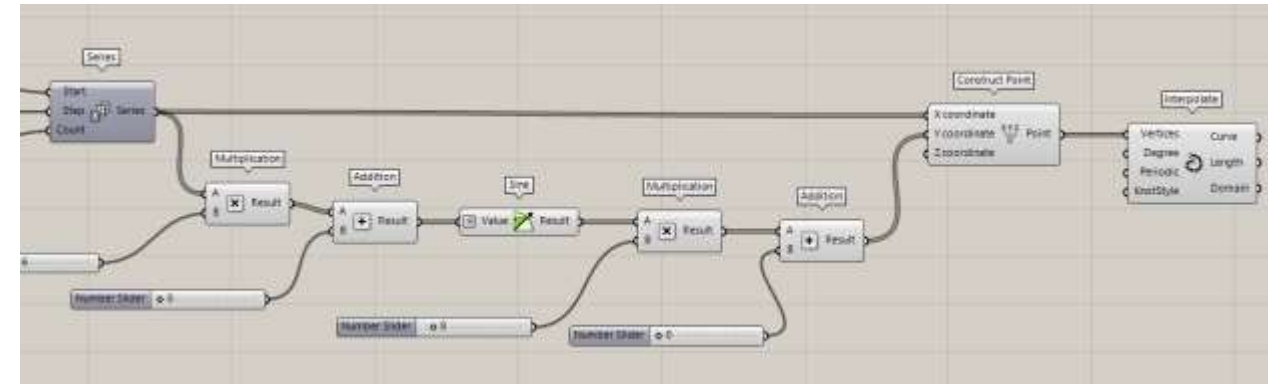
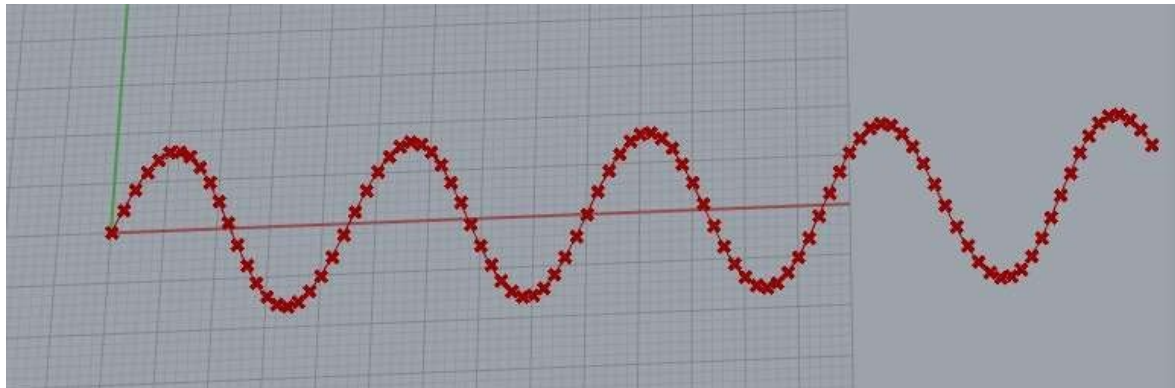
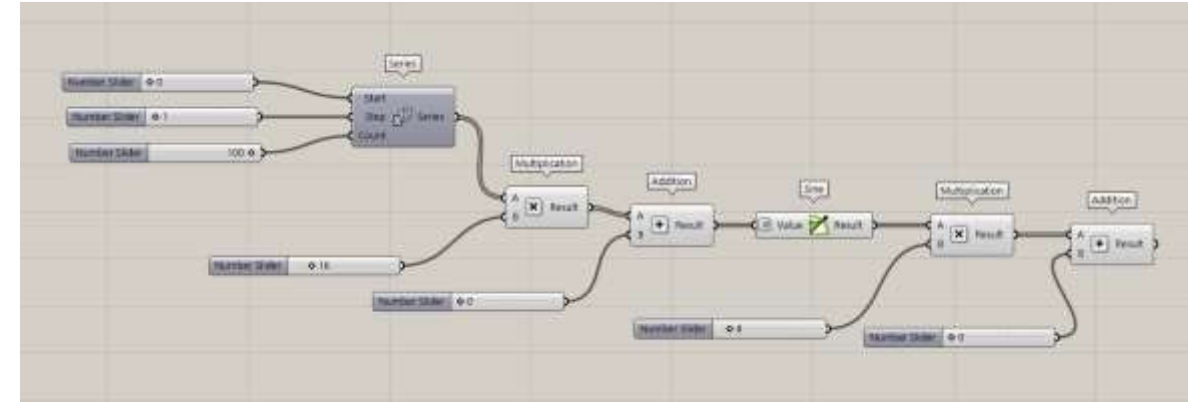


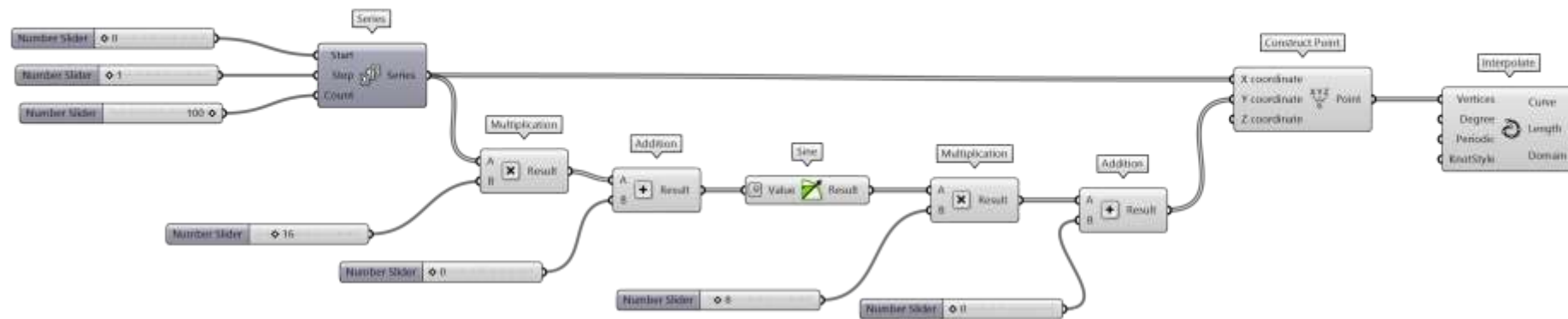
- E) Let's examine the result from node "sequence". To do so, we need to find node "Panel" (1). As we can see, we have the list of numbers. Those numbers will be used as X coordinates.
- F) To find out which numbers we will use for Y coordinate, let's find the SINE equation:  

$$Y=a*\sin(b*x+c)+d$$
- G) We need to multiply our series by d, add c and connect result to node called "Sine". By default, this node works with radians. To change working mode to degrees, you need to click right mouth button on sine unput slot and choose Degrees.

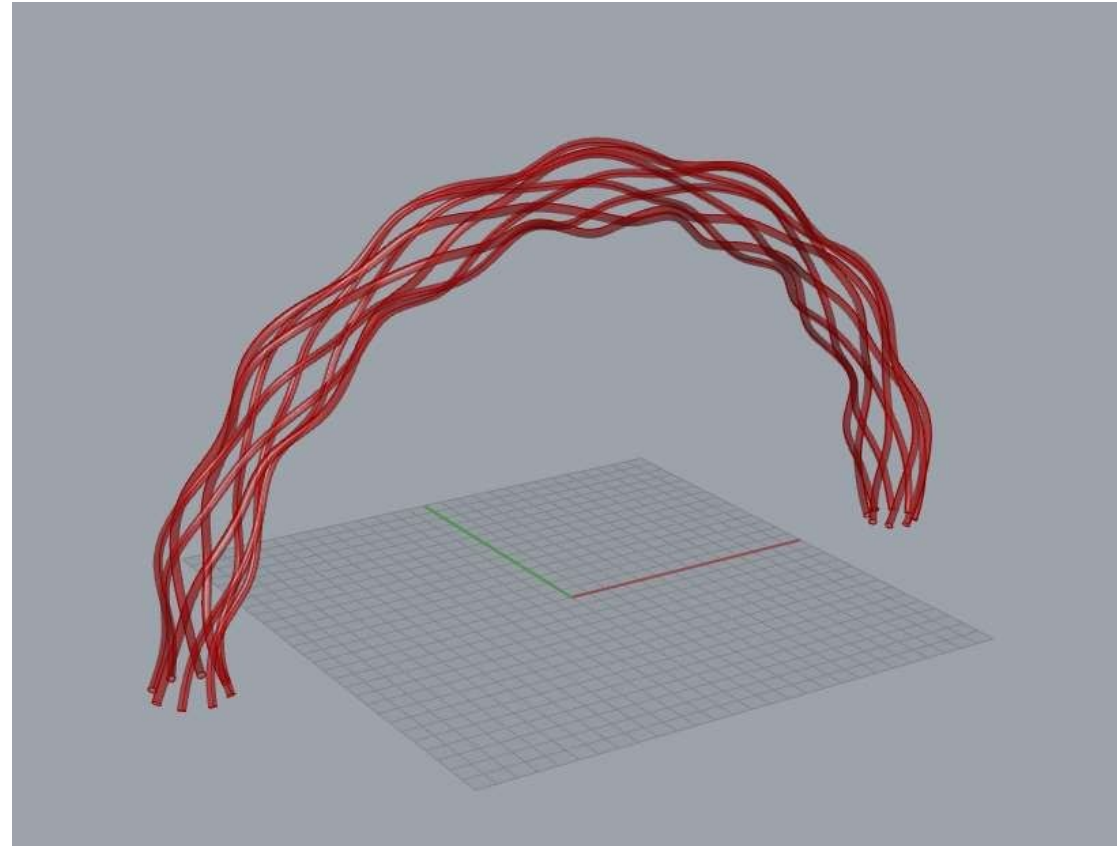


- H) According to the formula, sine should be multiplied and we need to add one more addition.
- I) Now let's add the node "Construct point". As an X coordinate we will use series and final addition as an Y coordinate.
- J) As a final node we will add "Interpolate" to make the curve by points.

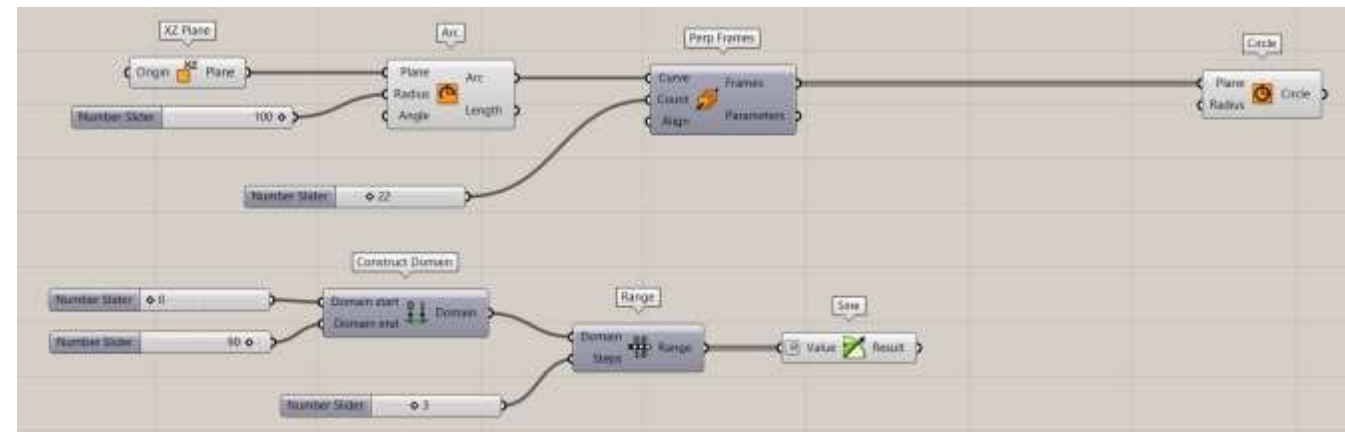
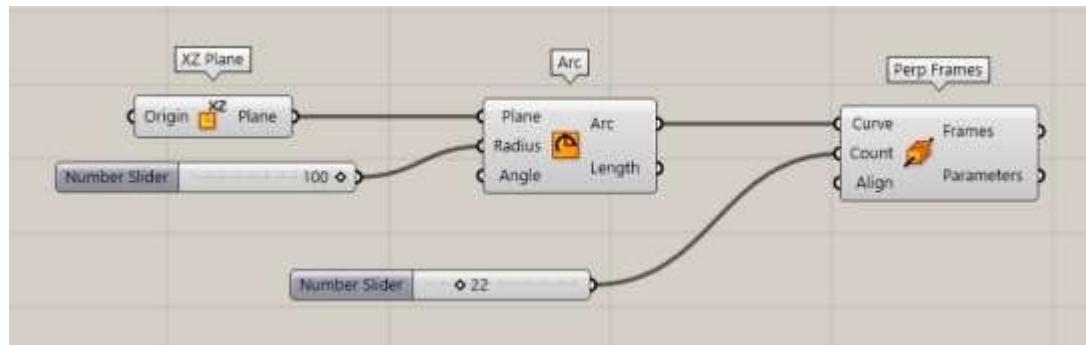




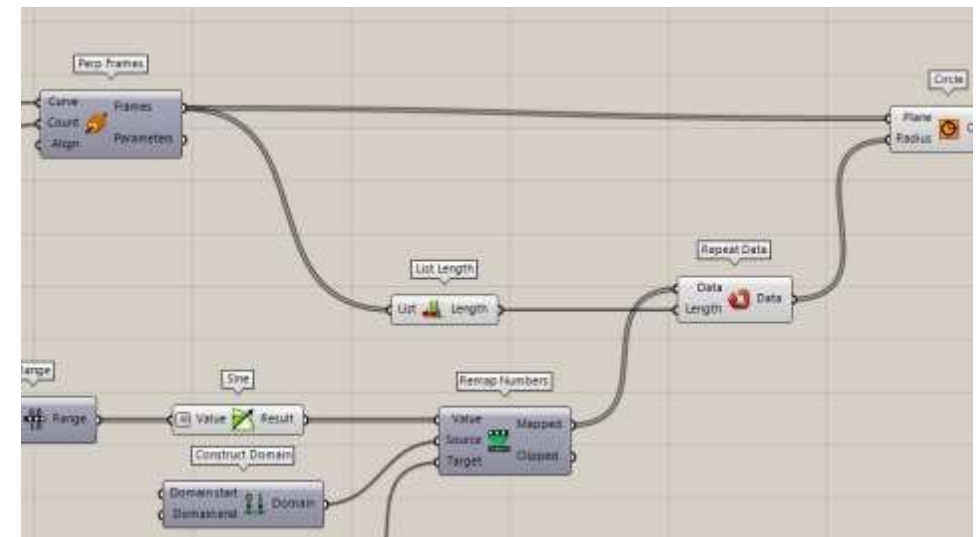
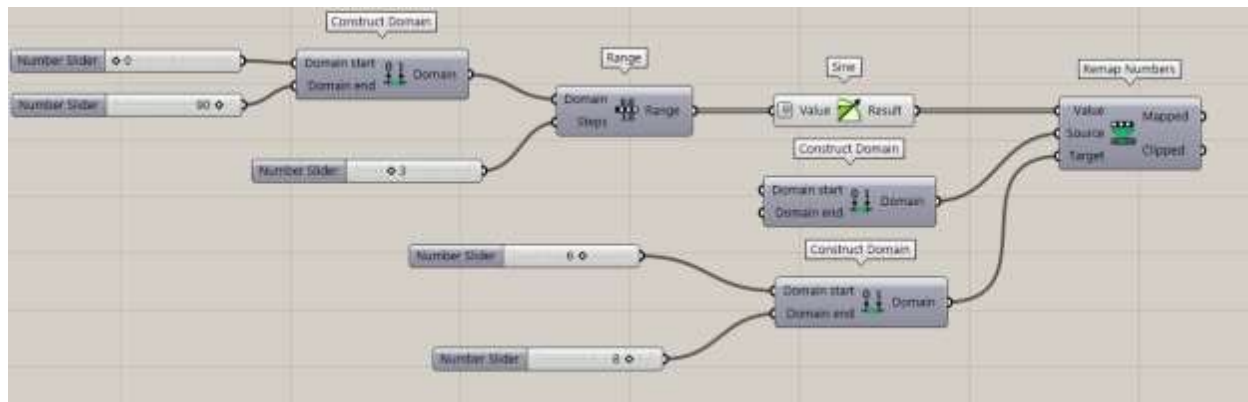
## Task 2 Parametric arc



- A) Call nodes “XZ Plan”, number slider and “Arc”. Connect them like shown on a picture (1).
- B) We will evaluate plans on the arc with the node “Prep Frames”. Plug arc on it’s curve input and add number slider for count (set integers). (1)
- C) Add node “Circle” and connect frames from “Prep Frames” to the plane input.
- D) Now we have set of circles, laying on the arc and oriented by it’s normal. For radiuses we will use SINE rule. To do so, let’s start our rule with node “Construct Domain”. Set domain start to 0 and end to 90 by two number sliders.
- E) Call the node range and connect domain with it and set amount of steps to 3 by number slider. It will give us four numbers: 0, 30, 60 and 90.
- F) Call node “Sine” and connect range with it and set the input value to degrees (2)

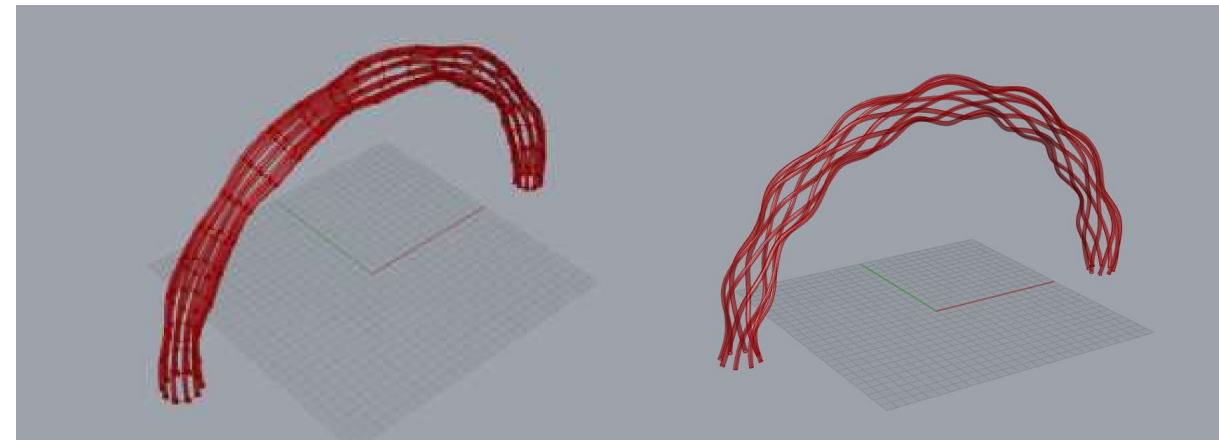
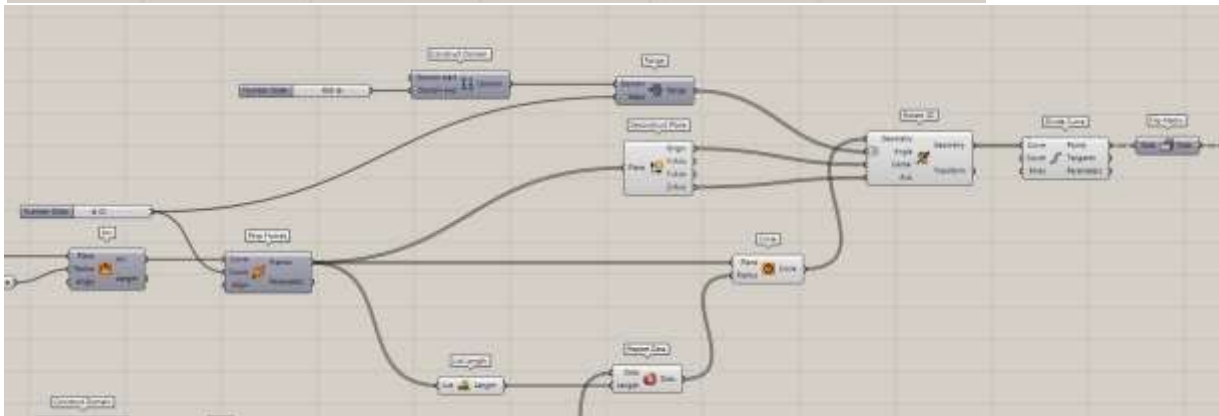
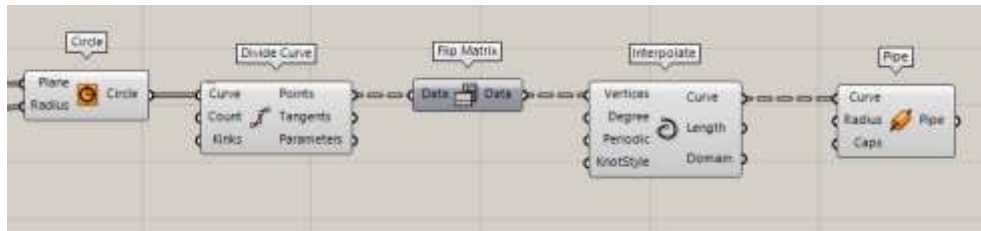


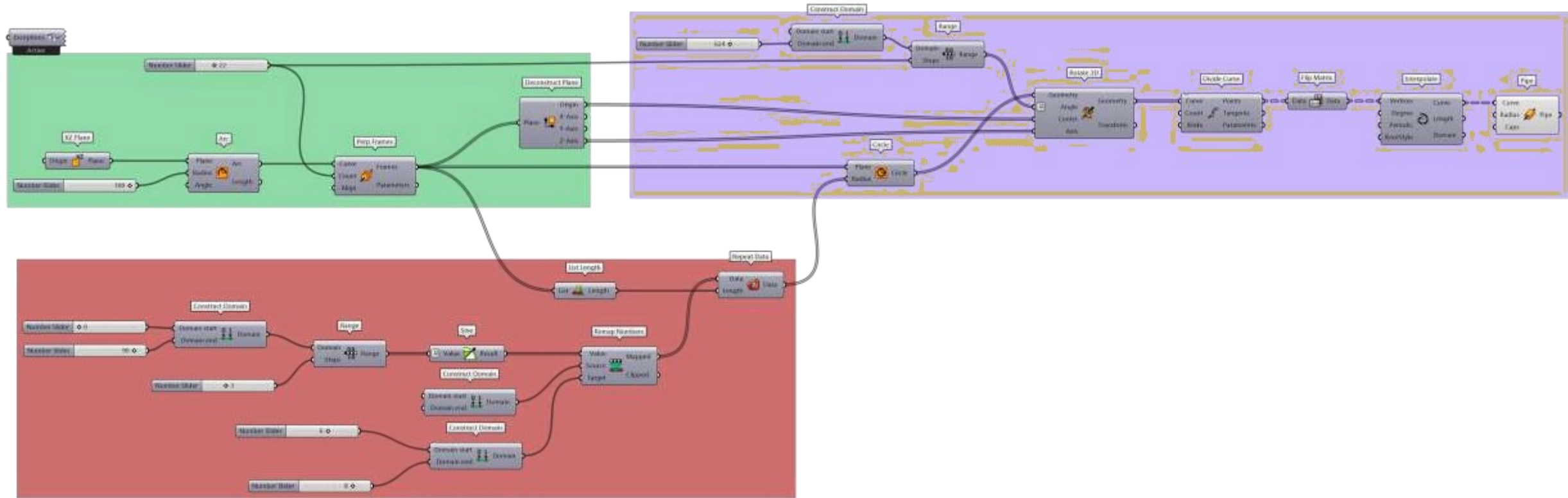
- G) Since the result of “Sine” node is too small, we will enlarge these numbers by using node “Remap numbers”. This node takes the list of numbers and changes their values with intervals being saved. Results of “Sine” are laying from 0 to 1. We want to convert them into 6 to 8 interval. To do so call two nodes “Construct Domain”. Leave one by default (since its default value is 0 to 1) and set domain start to 6 and domain end 8 for second domain (1).
- H) The result is 4 numbers, but we need more radii, since we have more circles. To match the number of radii to the number of circles let's use node “Repeat Data” and “List Length”. Connect “Prep Frames” to “List Length”, “List length” to “Repeat Data” into length slot and connect the result from “Remap Numbers” to data slot of “Repeat Data” (2).





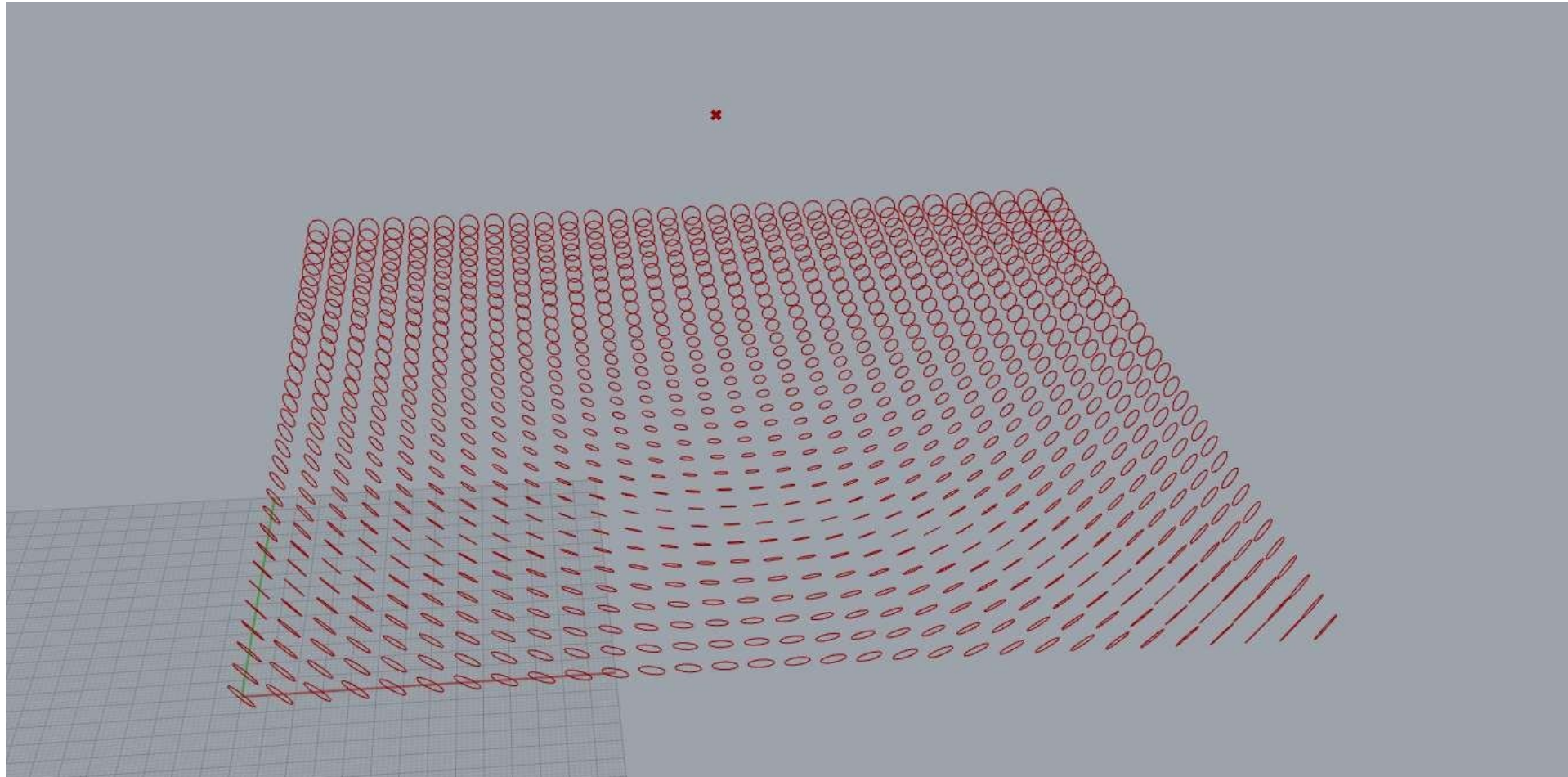
- I) Now we can divide circles by points, connecting node “Circle” to “Divide Curve”.
- J) “Divide Curve” gives us data tree with points. If we will try to make the lines from them, we will see same circles. To avoid this, connect points to the node “Flip matrix” than “Flip matrix” to “Interpolate” and “Interpolate” to “Pipe” (1).
- K) As we can see, the result is differs from the task picture (2). To correct this let’s get back to step H (do not delete nods from step I and J).
- L) To apply rotation you need to change script like it shown on the picture (3).



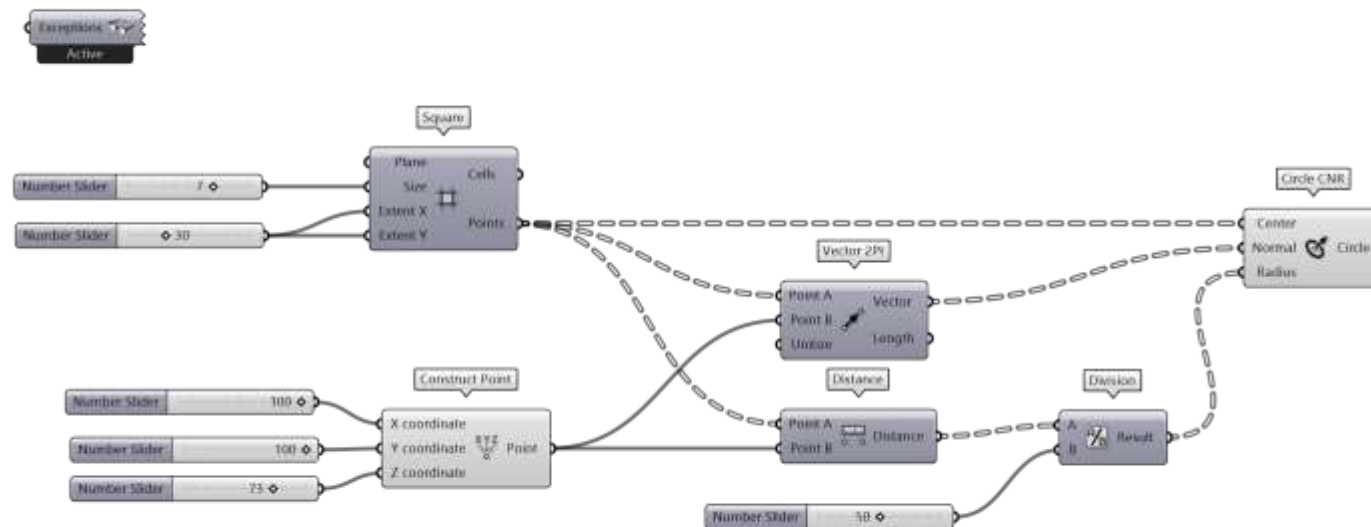




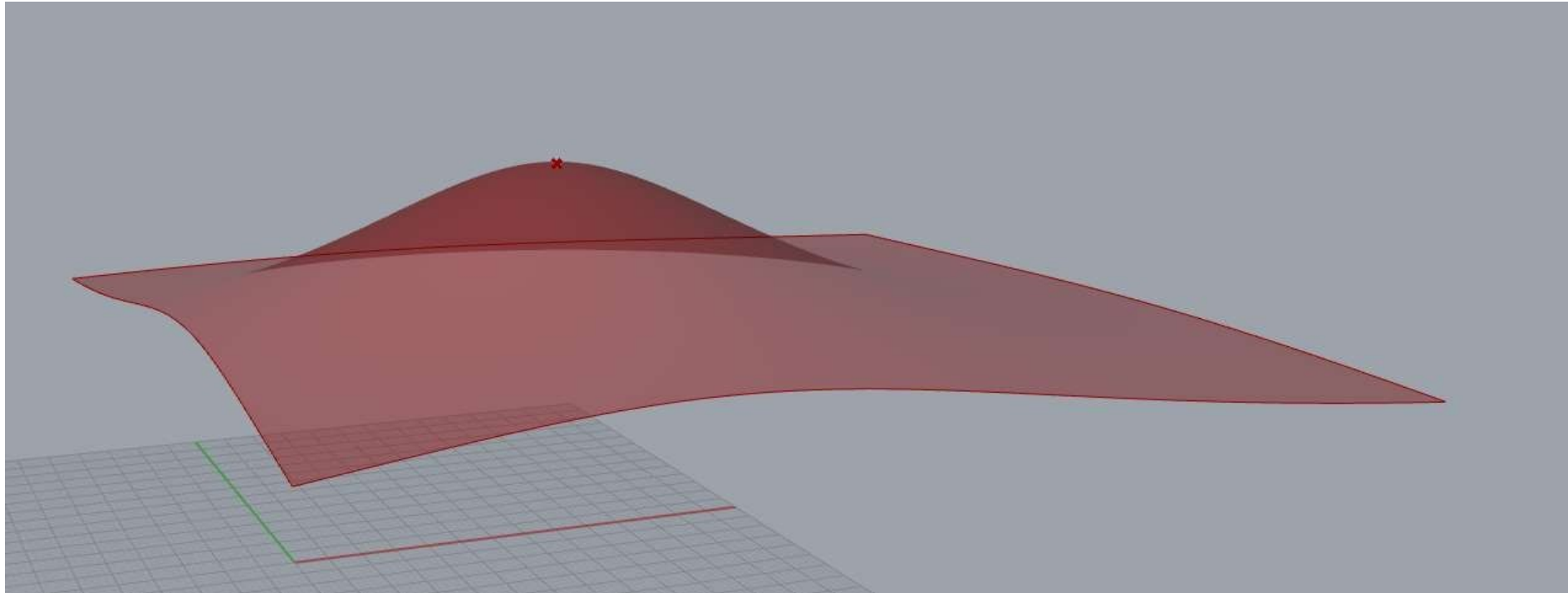
### Task 3A Attractor



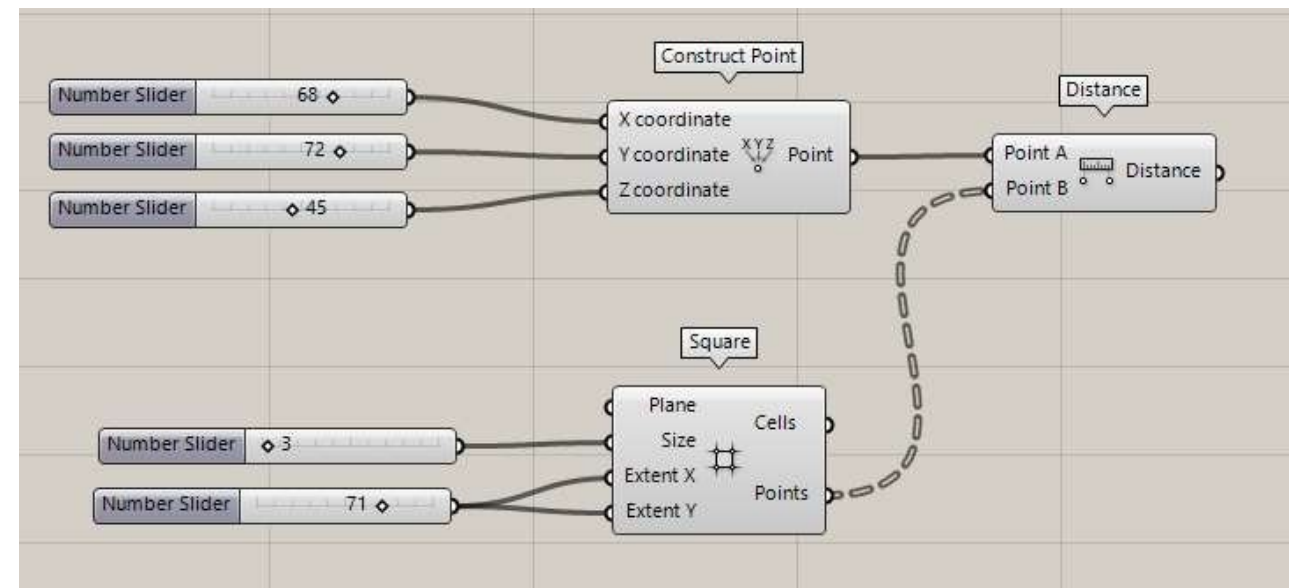
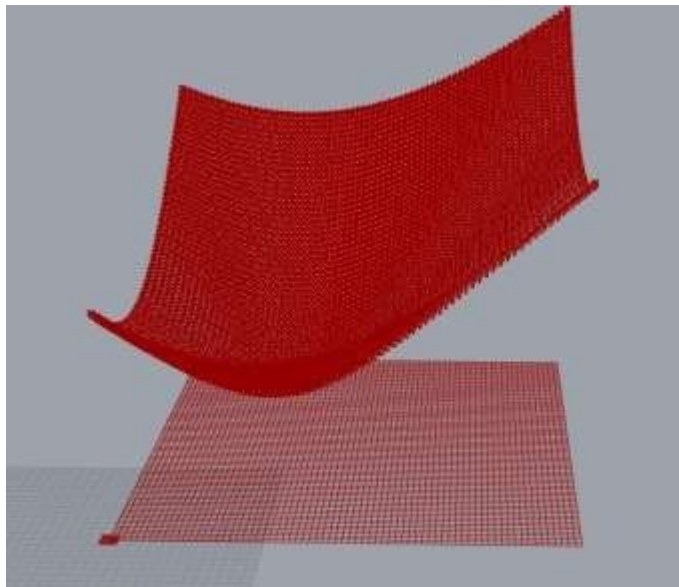
- A) Call node “Square”. Add two sliders. Connect first to size and second to both extent slots.
- B) Add node “Construct point” and make three sliders for each coordinate slot.
- C) Call node Circle CNR. Plug points from “Square” to center.
- D) To make normal we will use node “Vector 2Pt”. Connect points from “Square” to point A and point from “Construct Point” to Point B. Connect vector to the normal in “Circle CNR”
- E) For radius we will use node “Distance”. Points from “Sequence” to point A and point from “Construct Point” to point B
- F) The numbers from “Distance” are too big. To reduce them use node “Division”. Put distance to slot A and number slider to slot B.



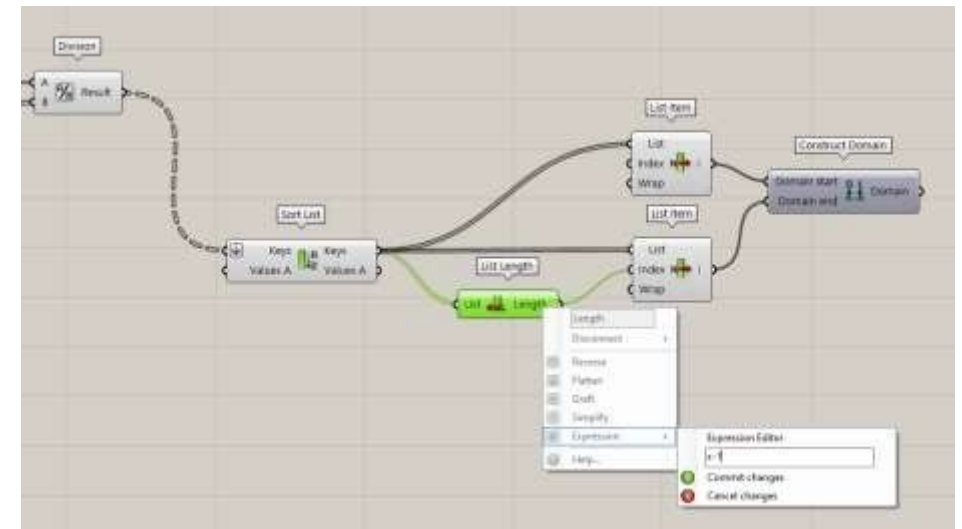
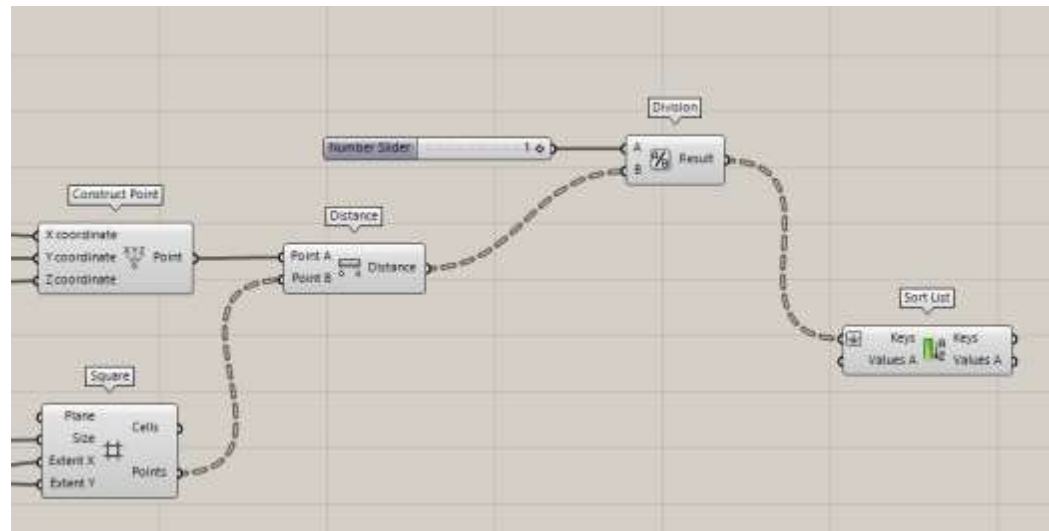
Task 3B Attractor (advanced)



- The idea of this script is to create surface, that will be ruled by one point. The difficult part is that we cannot use just the same concept like in previous task without modification, since using just a distance to move points will cause incorrect result (1).
- To do it right we need to apply sorting and transformation of distances, that will be used for points movement.
- A) Let's start with the set of points, point that will rule the surface (attractor) and distance between them (2).



- C) For proper translation values we will divide 1 by distances and sort them from lowest to biggest. To do so, we will use the node “Sort List”. Pay attention that if we will use it for tree, it will sort each lower list separately. And since we need lowest and biggest value, we need to apply flatten to the input slot of the node.
- D) Now it is sorted from lowest to biggest. To take first value from the list we will use “List item”. To take last value, use combination of nodes “List item” and “List Length”. “List Length” counts the items in the list. Pay attention that indexes in the list starts with 0, but list length gives the number that is bigger than last index by 1. To avoid this you need to add expression “x-1” to the input slot of “List Length”. To do so, click to the input slot, click to expression and type “x-1”.
- E) Make the domain from those two values (2).

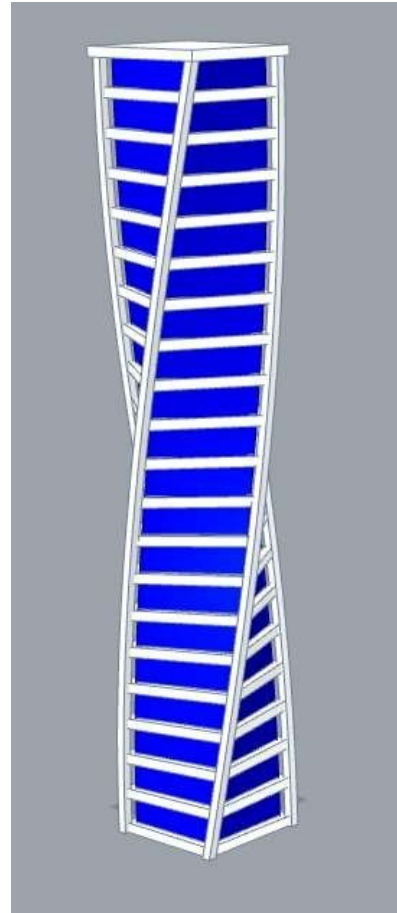






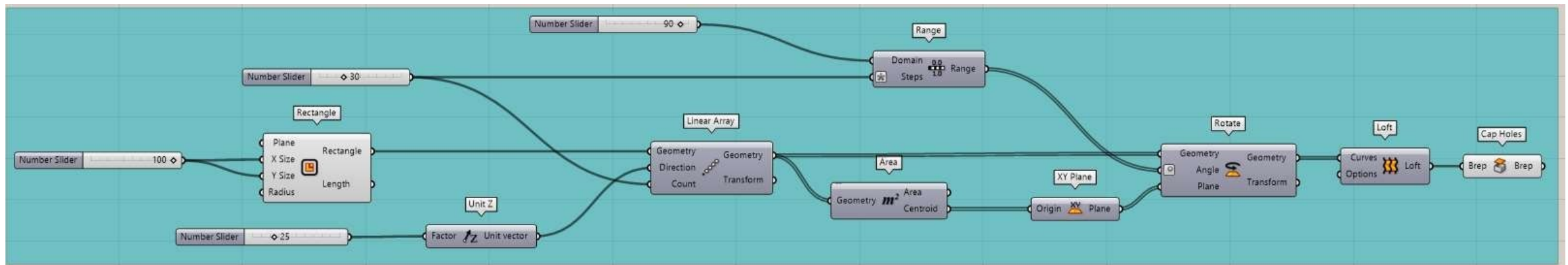


## Task 4 Parametric tower

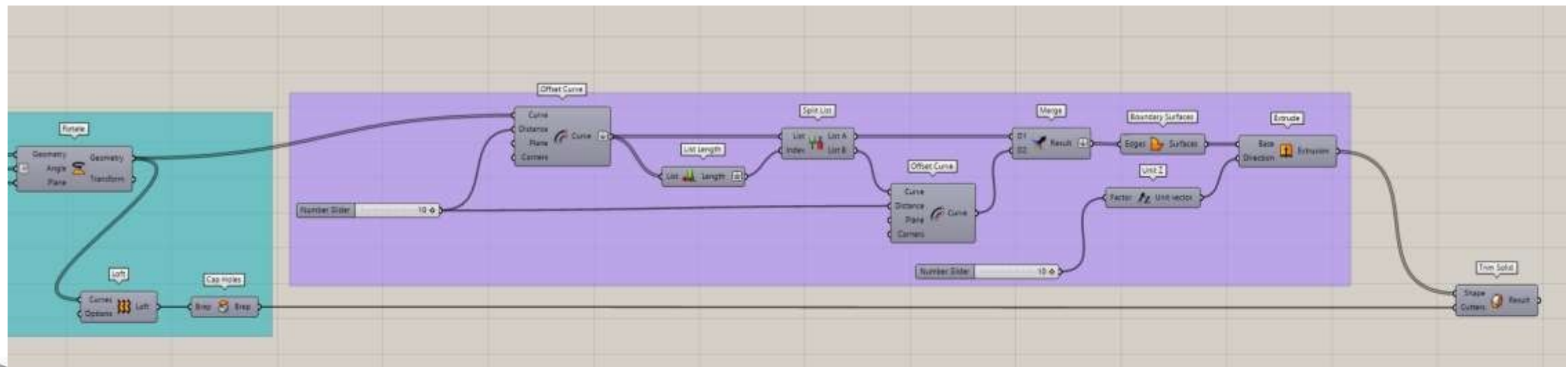




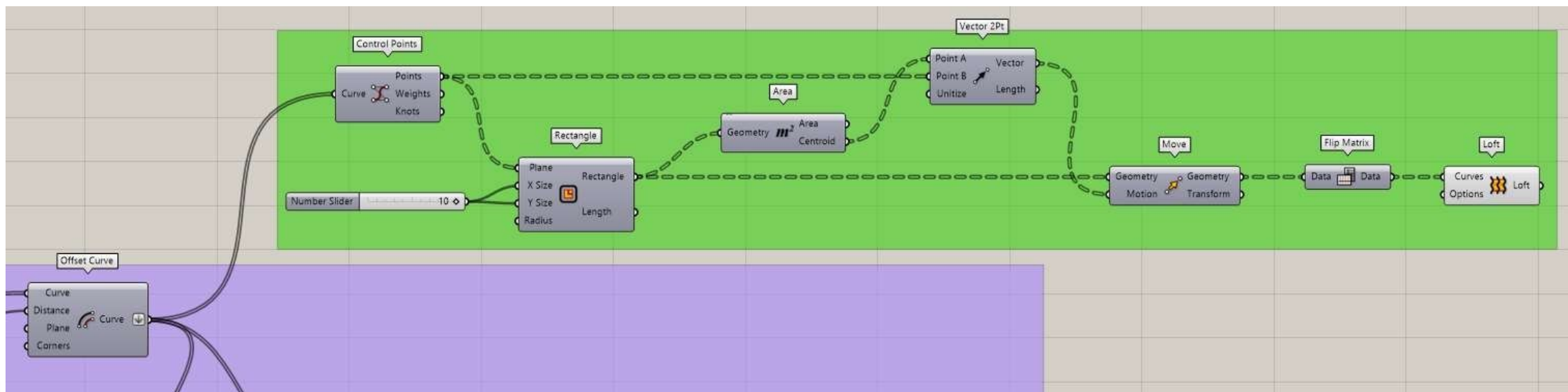
- A) Call node “Rectangle” and set X and Y size to 100.
- B) Call node “Linear Array”. For direction use “Unit Z” and set number of array elements to 30.
- C) To apply rotation we will use “Rotate”. This node requires plans that located in the centre of rotation. To find centre of 2d geometry, use node “Area” and make XY plan from it. For angle we will use the “Range”. Use number slider instead of domain (it will create domain {0 To n}) and for steps use slider that we used for array before (set expression {x-1}). Do not forget to set degrees into angle input.
- D) To make solid structure from this sections use node “Loft” and “Cap Holes”

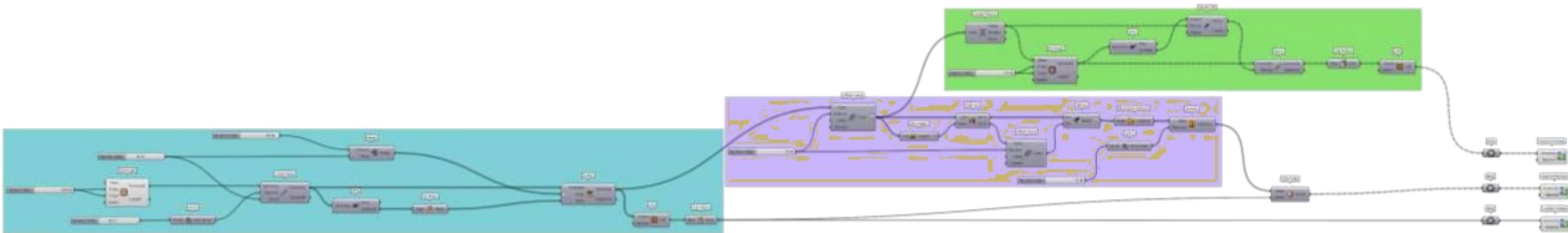


- E) Now we will make horizontal parts. Call node “Offset Curve”. Connect rotated sections from “Rotate” and set distance to 10.
- F) To make top horizontal part wider, we will split the list of offset curves by “Split List”. Top part is the final element of list, that’s why we will use “List Length” (with {x-1} expression). In list B input we have this part and rest in list A.
- G) Use “Offset Curve” one more time for top and connect back two lists by node “Merge”(add flatten to output slot).
- H) To create 3D geometry from line sections use “Boundary Surfaces” to make surface and “Extrude” to make brep. Use “Unit Z” for direction.
- I) Add node “Trim Solid” to cut horizontal sections.

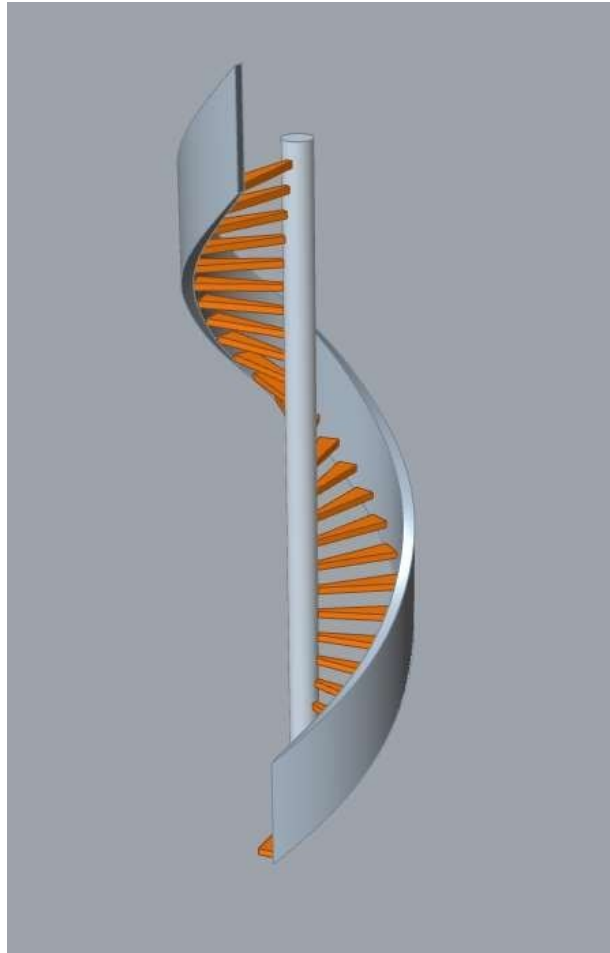


- J) To make vertical parts we will use control points from horizontal sections by “Control Points”
- K) Make rectangles from control points. Plane in “Rectangles” is used to adjust corner of the geometry (not centre), but in this case we need to change rectangles position to centre. To do so, use nodes “Area”, “Vector 2Pt” and move like it is shown on the picture.
- L) Use “Flip Matrix” to give proper structure for the node “Loft”.
- M) To add colours for geometry, call three nodes “Custom Preview”. Plug “Cup Holes”, “Trim Solid” and “Loft” into “Custom Preview”. Right click to the material slot and choose “manage shader collection”. Click on “Add Item” and choose preferred coloures.

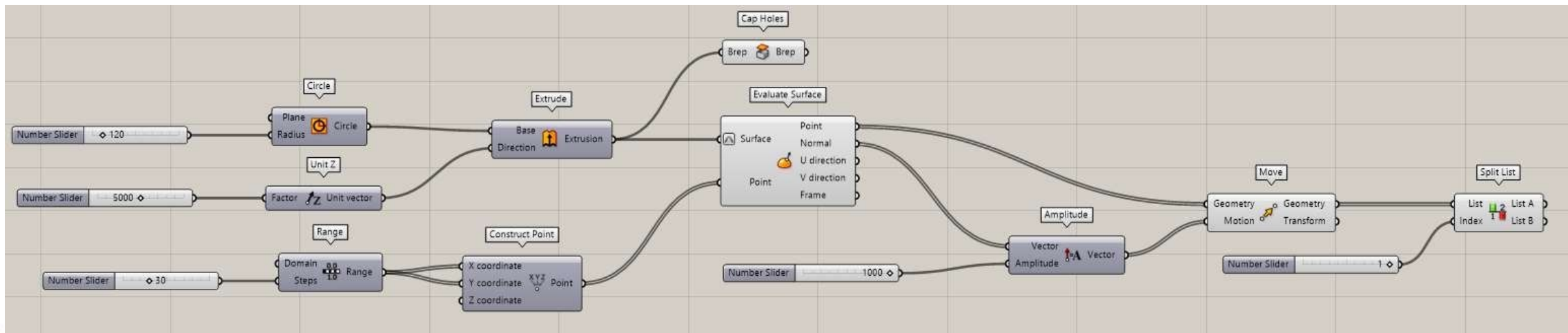




## Task 5 Parametric staircase

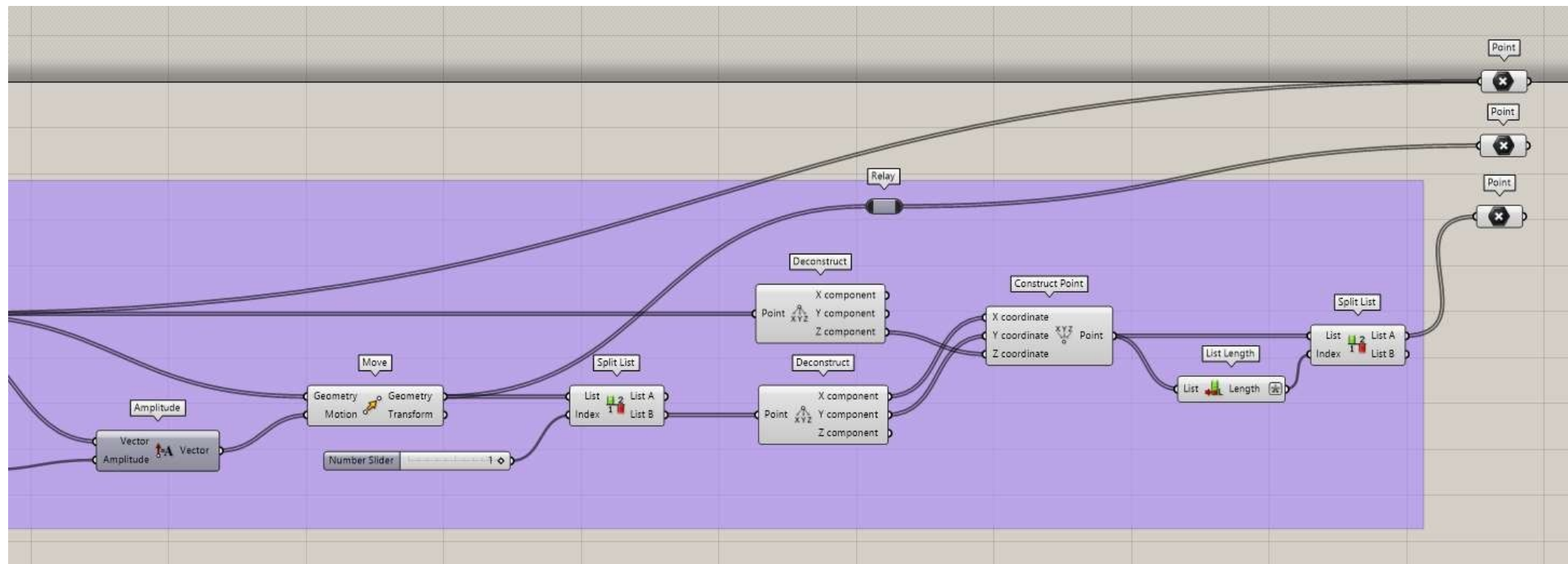


- A) Make circle and add number slider for radius.
- B) Call “Extrude”. “Unit Z” plug into direction slot. Add slider for Z factor.
- C) Use “Cap holes” for the pipe.
- D) Call node “Evaluate Surface”. Add reparametrize for surface slot and plug pipe there. For point we will create set of points in UV coordinates. To do so, call node “Construct Point” and make set of points (Set of points № 1) with coordinates {0 To 1, 0 To 1, 0}.
- E) To make three needed sets of points we will use points and vectors (normal) from “Evaluate Surface”. Use node “Amplitude” to enlarge vectors length (normal from “Evaluate Surface” have 1 length) and plug it with points into “Move” (Set of points № 2).
- F) To make Set of points № 3 we need to get rid of first point after “Move” by using split list.

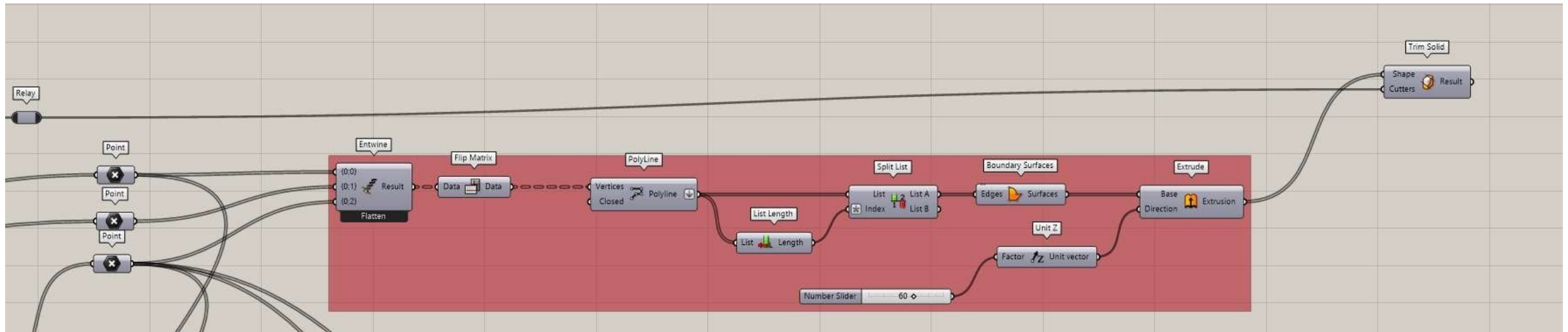




- G) Deconstruct point from list B and deconstruct points from set №1. Construct new points from XY coordinates from List B and Z coordinates from set №1.
- H) Get rid of the last point by “Split list” and “List Length” (with  $\{x-1\}$  expression). It is set of points №3.
- I) Call three “Point” nodes and connect with each set of points (for easier access later on).

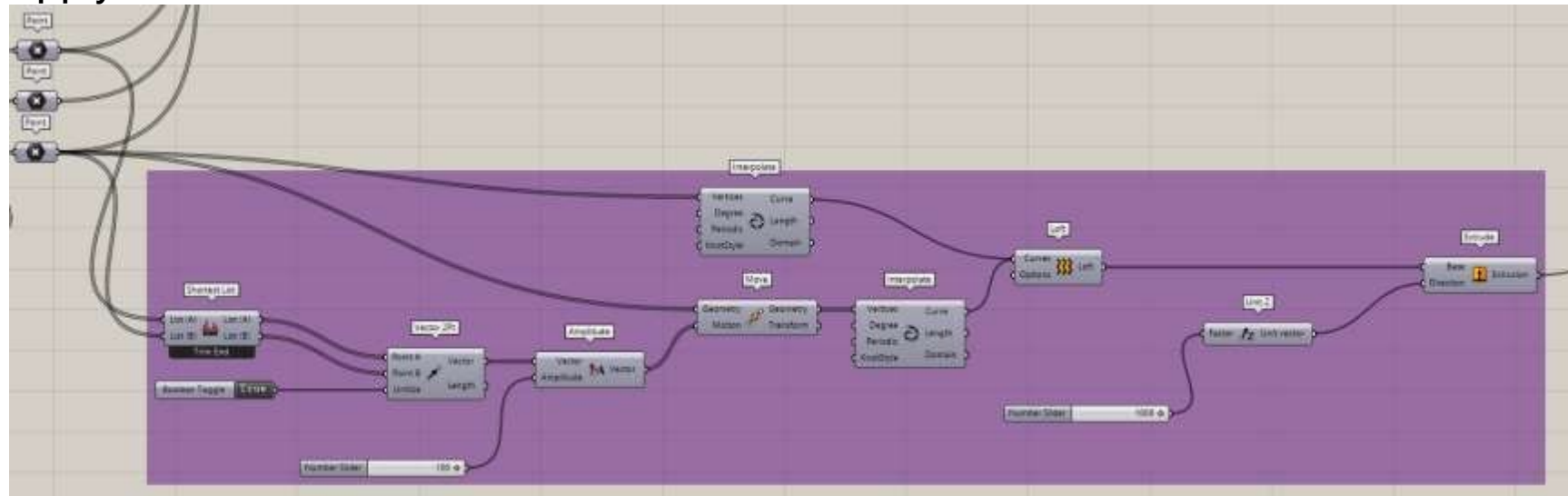


- J) To make stairs we will combine all three point sets into one tree by using node “Entwine”.
- K) Flip this data tree by “Flip Matrix”
- L) Make polylines from those points
- M) get rid of the last polyline by “Split List” and “List Length”
- N) Make surfaces from polylines by “Boundary Surface” and make breps by “Extrude” and vector Z.
- O) To cut parts inside the column use the node “Trim solid” (“Cap Holes” for cutters).

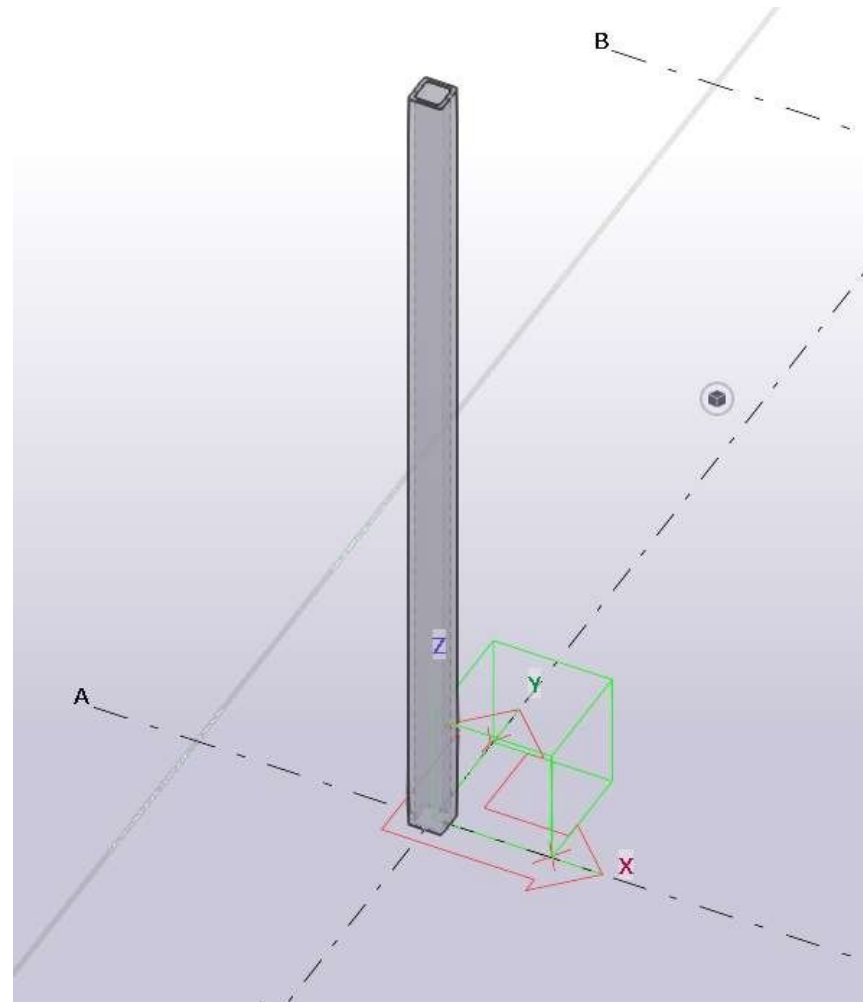




- P) To make fencing let's call "Shortest List". Plug set of points 1 into list A and set 3 into list B
- Q) Call "Vector 2Pt" List A and List B from "Shortest List" to point A and Point B. Call Boolean Toggle, set True and plug into Unitize.
- R) Call "Amplitude" and set amplitude to 100.
- S) Call "Move". Plug set of points 3 into geometry and "Amplitude" to the motion.
- T) Call two nodes "Interpolate". Plug set of points 3 into first node and move into second.
- U) Call "Loft". Plug curves from both interpolate nodes into curves slot (to plug both use Shift while dragging). Extrude this surface into Z direction.
- V) Apply colours for all three solids with "Custom Preview".



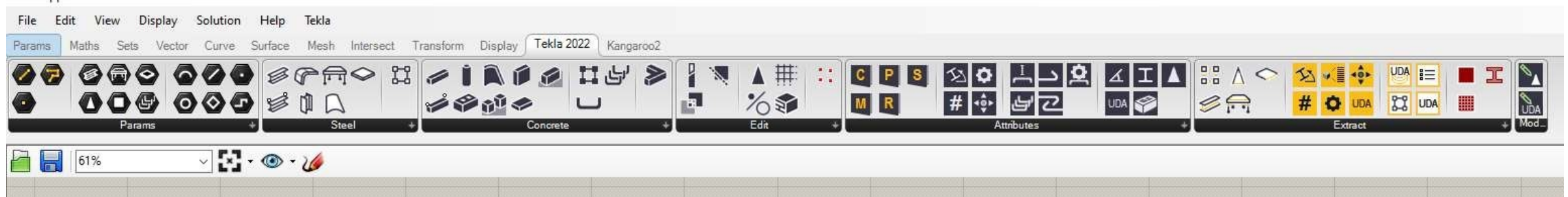
## Task 6 Connection with Tekla



*BIM-ICE – BIM-Integration in Higher and Continuing Education (KS1905)  
Co-funded by the European Union*



- To connect Grasshopper with Tekla:
- A) In Grasshopper press File-Special Folders-Components Folder and move plugin files there.
- B) Right click on the file .gha and press Unlock
- C) Close Grasshopper and Rhino.
- D) Open Tekla and create new file
- E) Open Rhino and Grasshopper. If you did everything right, you will see new node page (Tekla 2022).



- F) To make custom column let's start with cross section (1).
- G) Make line that will be used as a height (2).
- H) Create profile from cross sections (3).
- I) Make a column from line and profile.

