C# in GH - Function Composability

Tokyo AEC Industry Dev Group - 06/13/2020

Ping-Hsiang Chen

Compose - ability / -possibility

In computer science, “composability” is a system design principle that deals with the inter-relationships of elements. “Function composition” is a mechanism which allows the result of one argument passed to another one to form a more complex one. The final outcome is the accumulative result of the entire process.

This formation concept resonates with the geometric system typically employed in architectural design and modelling procedure.

In this session, the emphasis will be placed on how geometric functions in Grasshopper C# can be composed and have interchangeability to another one. We will also investigate how to create shape grammar based on the principle introduced.

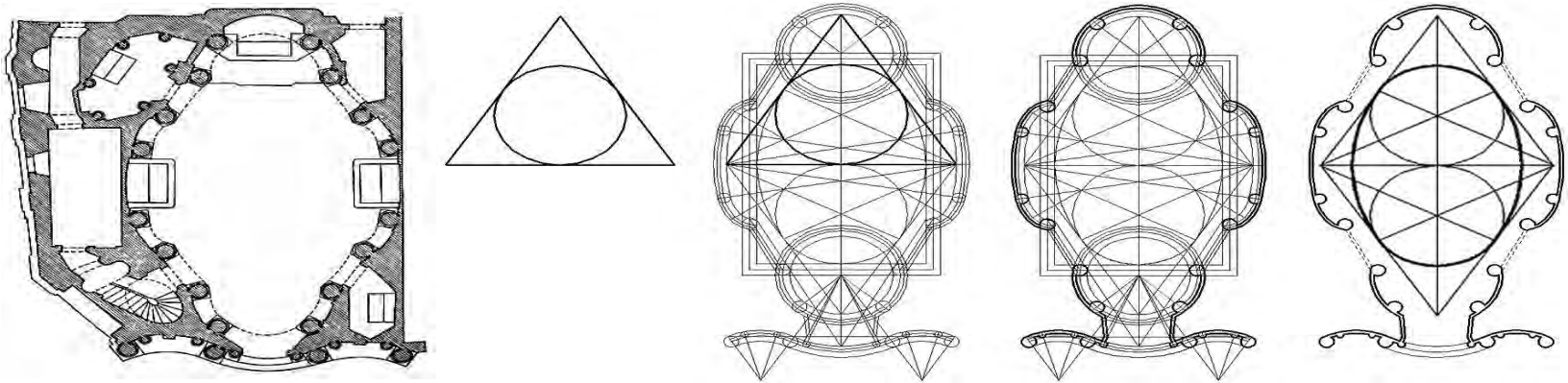
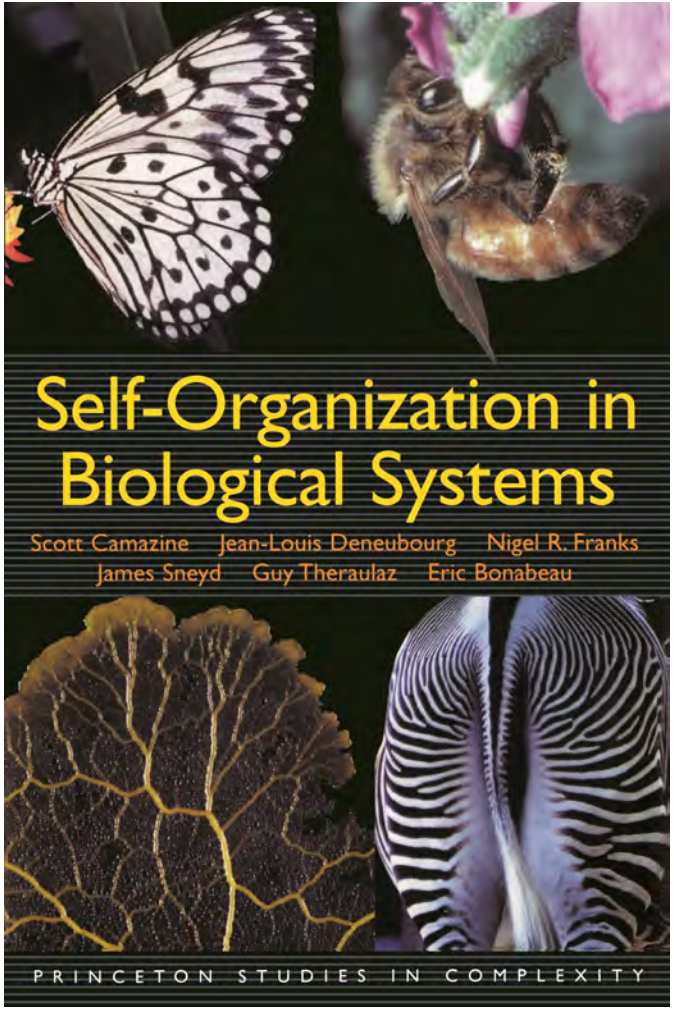


image source: Google

https://digital.kenyon.edu/cgi/viewcontent.cgi?referer=https://www.google.com/&httpsredir=1&article=1061&context=perejournal



Topics

1. Function Handling in RhinoCommon
2. NodeInCode Function & Use Cases
3. Grasshopper Data Structure
4. Geometric Composition

Related Topics

1. Emerging Behaviour
2. L-System
3. Shape Grammar

Software

Rhino 6.0 & plug-ins
(https://drive.google.com/open?id=16SBwSGzBacYTckU6d1tvKlzk2k6R7crT)



Overview

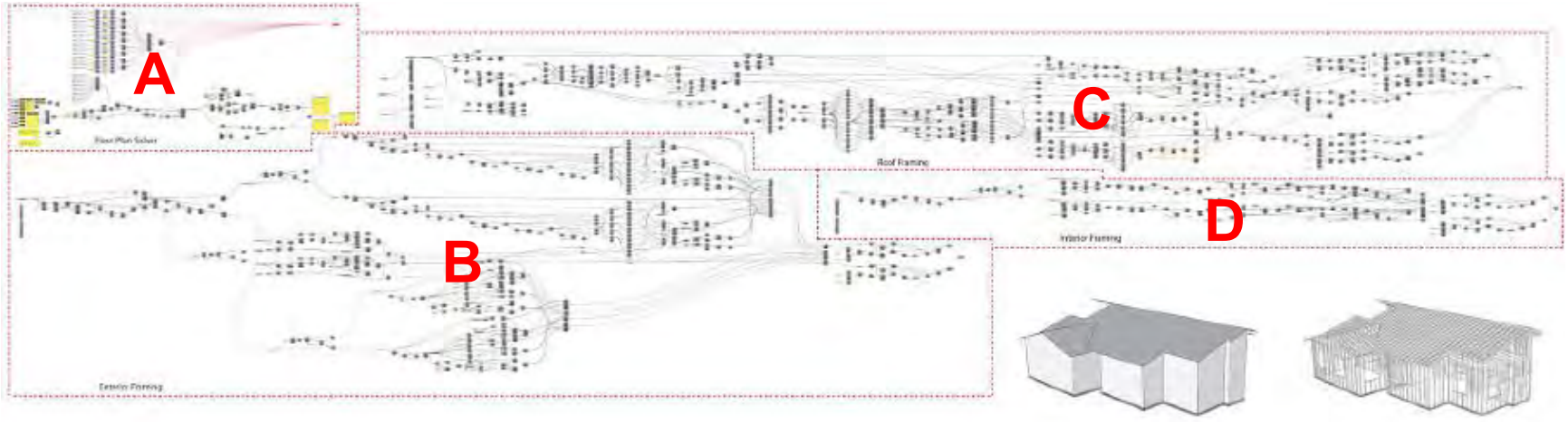


image source: Google

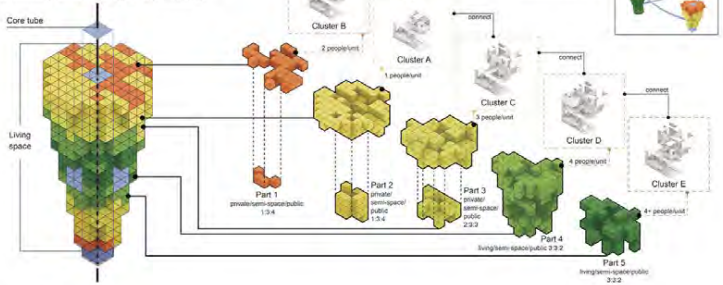
UNDERGROUND X-Community 3

Workshop 2, Group 4

Group members
Daoyuan CHEN, Pengcheng ZHAI, Zheng CHEN, Zheng CHEN, Kaijie YU, Yize LIU
Tutor
Ping-Hsiang Chen, Peng Qin

Community Explosion Diagram

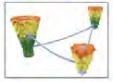
An example by Type 3
The relationship between clusters, parts and communities



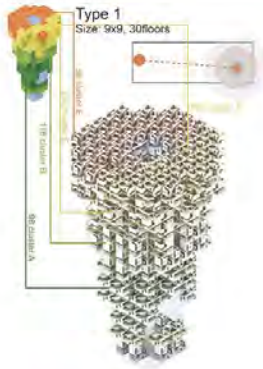
Community Size

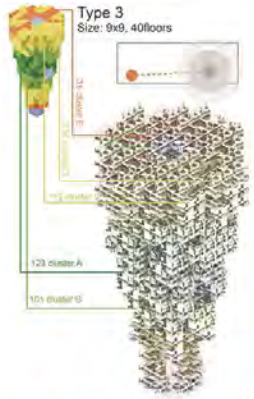
The distance between the community and the resource point control the size of the community. The closer the resource points are, the larger the size of the community, which means the community has more clusters and more layers.

With different types, communities that depend on the same resource point will be interconnected to form a group.



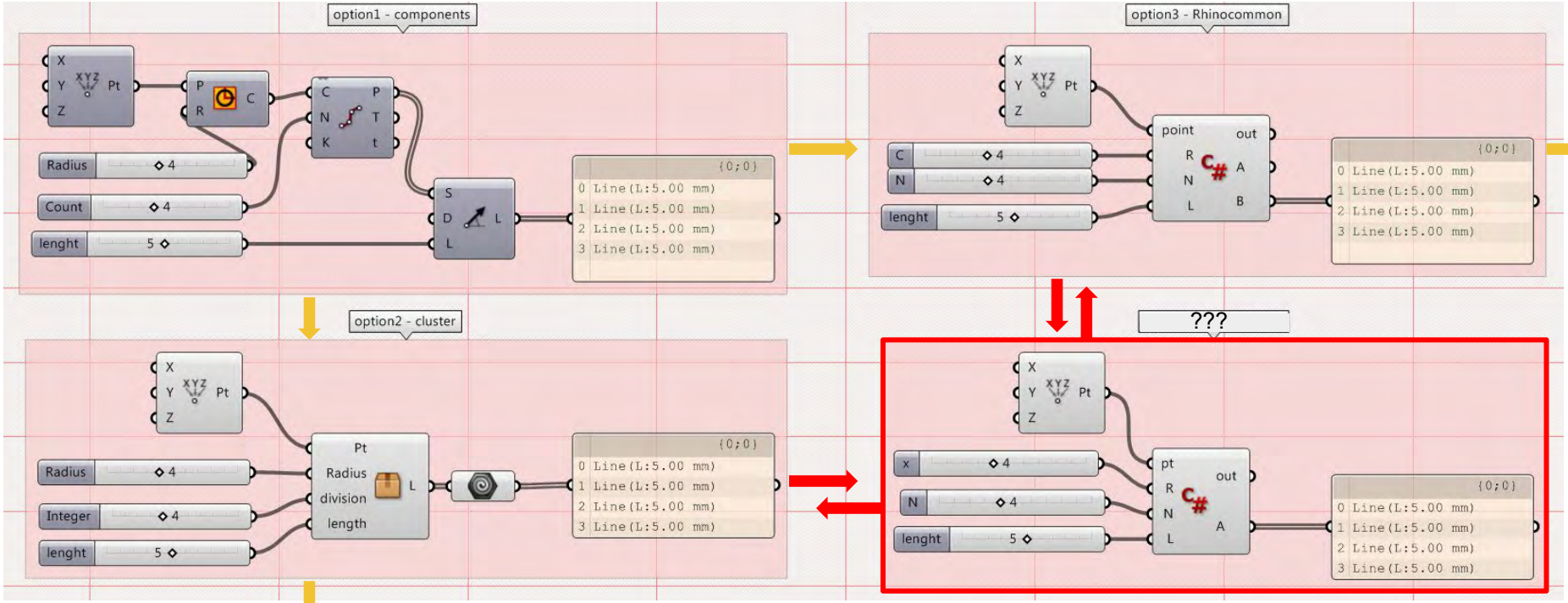
Community Type



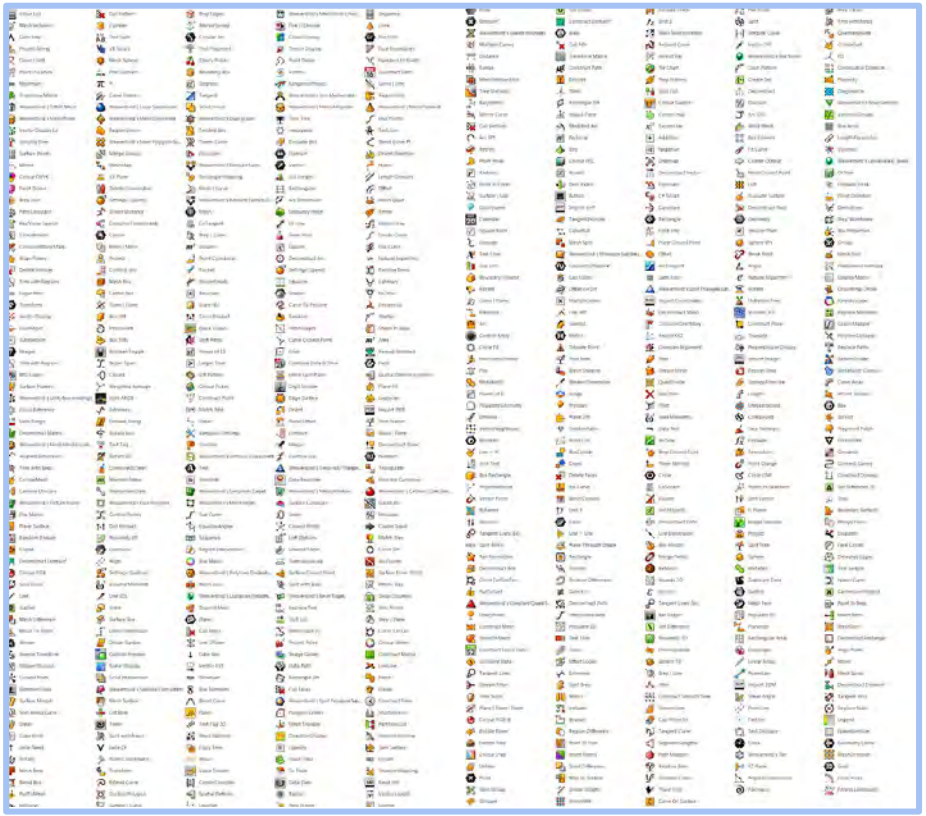


Overview

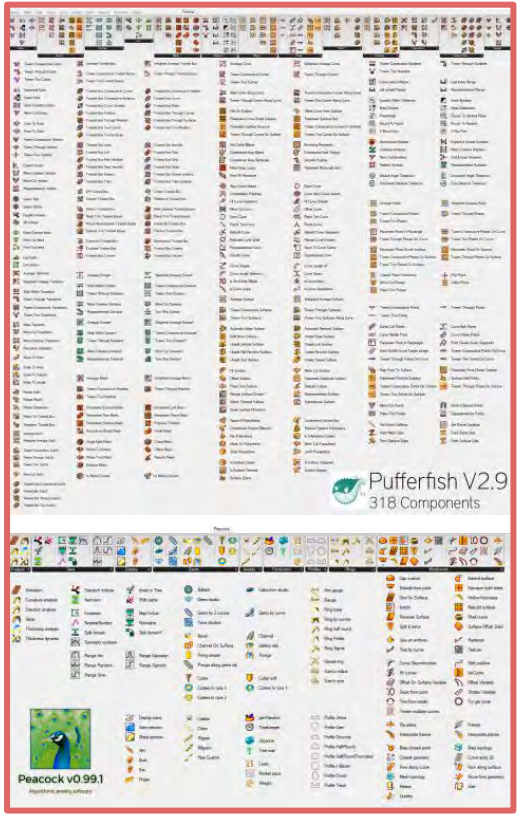
- Function composition
- Code re-usability
- Function extension
- Iterative capacity



Overview



Grasshopper functionality

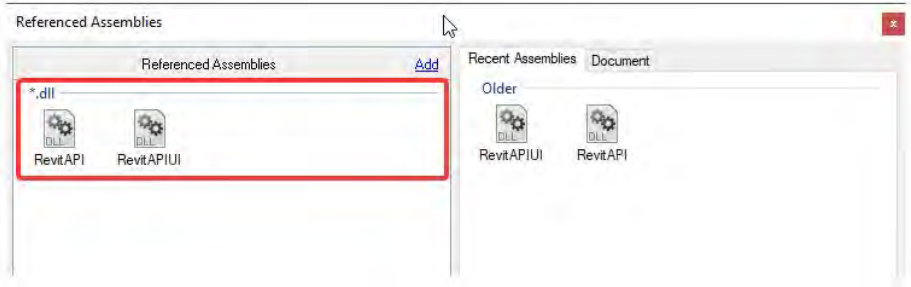


Plug-in functionality

image source: Google

Overview - function extension

Assembly management



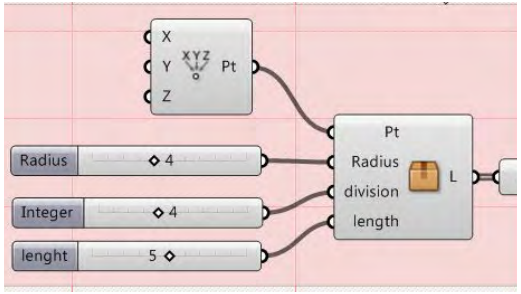
Syntax management

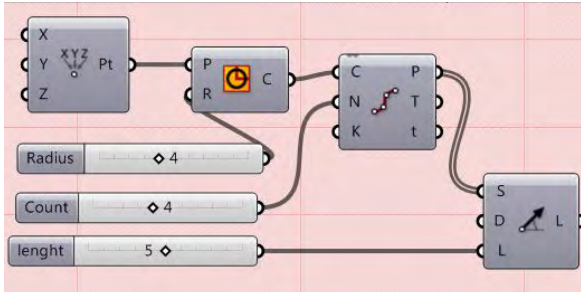
Namespace management

```
using RIR = RhinoInside.Revit;
using DB = Autodesk.Revit.DB;
using UI = Autodesk.Revit.UI;
```

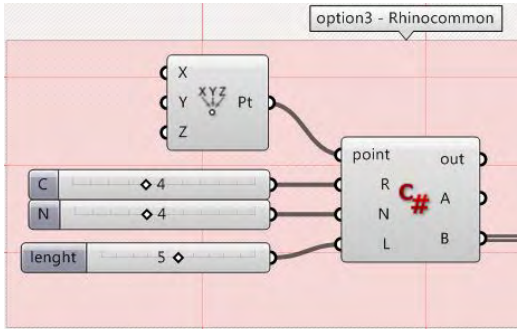
Interoperability Issue

Cluster vs Rhinocommon





GH geometry / data type principle



Base and Derived class

Curve
ArcCurve
CurveProxy
LineCurve
NurbsCurve
PolyCurve
PolylineCurve

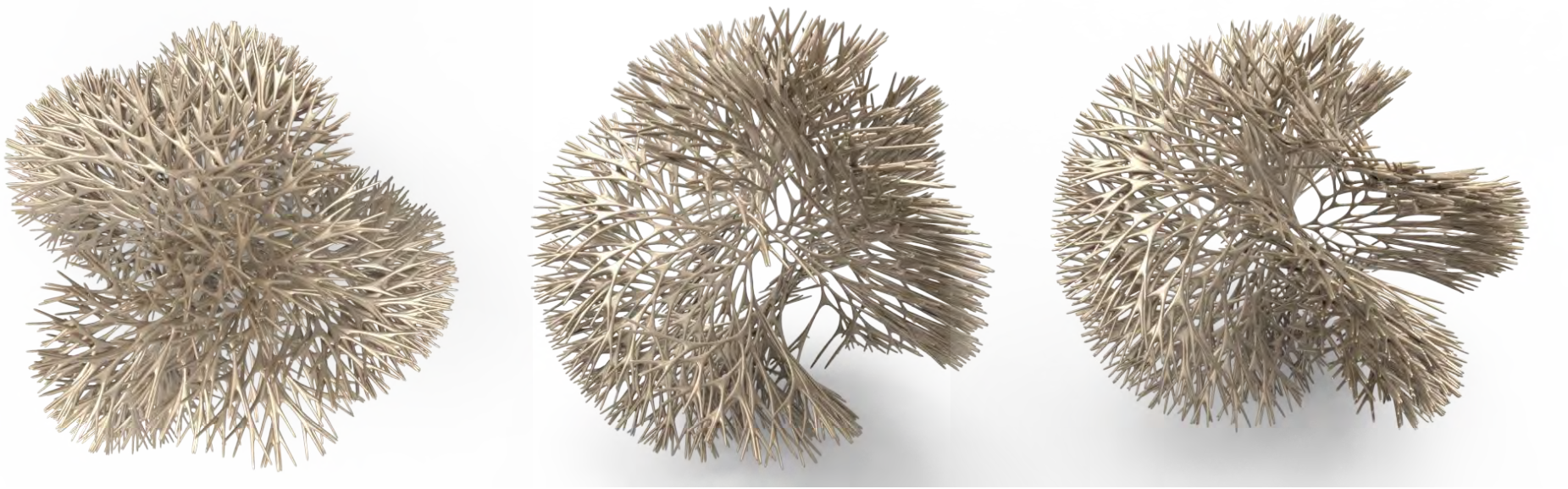
Ex. PointAt(t)

Overload methods

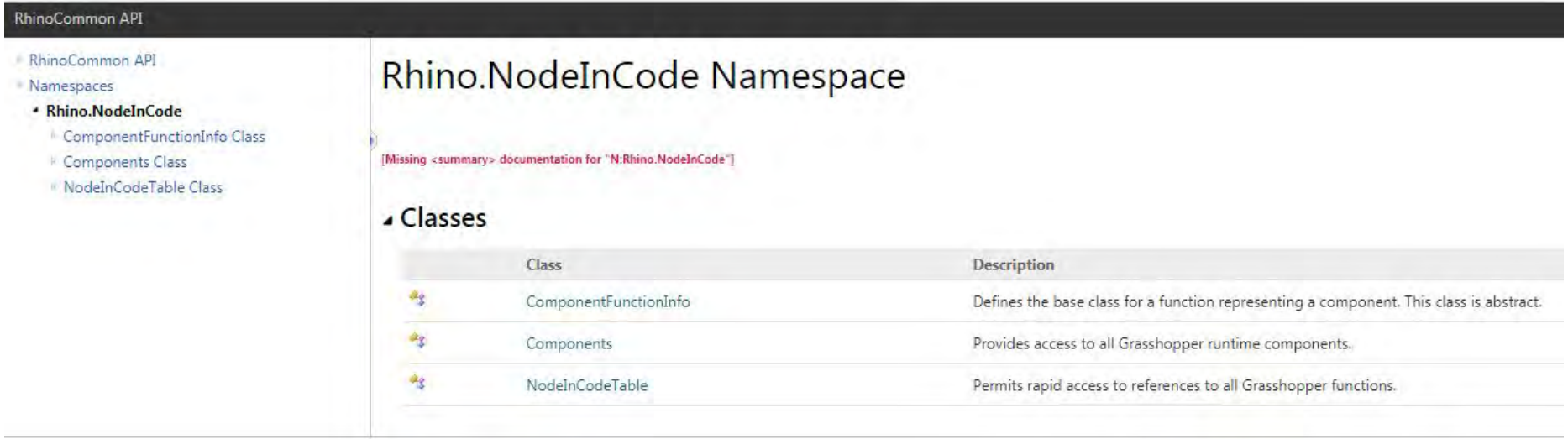
Point3d(Point3d)
Point3d(Point3f)
Point3d(Point4d)
Point3d(Vector3d)
Point3d(Double, Double, Double)

(Explicitly define)

Iterative Capability



NodeInCode



https://developer.rhino3d.com/api/RhinoCommon/html/N_Rhino_NodeInCode.htm

Reference

https://developer.rhino3d.com/guides/rhinopython/ghpython-call-components/
https://stevebaer.wordpress.com/2013/12/11/ghpython-node-in-code/
https://discourse.mcneel.com/t/using-the-nodeincode-namespace/49735
https://discourse.mcneel.com/t/new-version-of-ghpythonlib-components/34619

NodeInCode

December 11, 2013 / stevebaer

ghPython – New component and parallel modules

Just in time for Christmas... ghPython 0.6.0.3 was released this week and it has two new features that I'm really excited about.

Download ghPython...

A little background

David Rutten was visiting the McNeel Seattle office in November to discuss future work on Grasshopper and Rhino. When David is in town it always gives me the chance to brainstorm with him and try to solve some of the features that users ask for. Two features that we commonly hear about are "how can I do what X component does, but through RhinoCommon/code?" and "how can I improve performance on my computer with many CPUs?"

Out of these chats came the two major new features in ghPython 0.6.0.3; the ability to call components from python and an easy way to do this using multiple threads. ghPython 0.6.0.3 ships with a new package (ghpythonlib) that supports these two new features.

Components As Functions (node-in-code)

There is a module in ghpythonlib called components which attempts to make every component available in python in the form of an easy to call function. Here's a sample to help paint the picture.

```
1  import ghpythonlib.components as ghcomp
2
3  # call Voronoi component with input points
4  curves = ghcomp.Voronoi(points)
5  # call Area component with curves from Voronoi
6  centroids = ghcomp.Area(curves).centroid
```

Rhinoceros Forums

Do you want live notifications when people reply to your posts? Enable Notifications .

Using Rhino remotely during COVID-19 • See what's in the Rhino 7 WIP

nodeincode

+ New Topic

33 results for **nodeincode**

Sort by Relevance

Using the NodeInCode namespace
Rhino Developer ghpython rhinocommon ghpythonlib grasshopper nodeincode
Nov '17 - ...lib-components/34619 ghpythonlib.components wrapper library , which is easier to use in Python. As some noticed some time ago, we have been working on **NodeInCode** functionality. The goal was to make this functionality more understandable – with warnings that could give useful suggestions, stable – also when use...

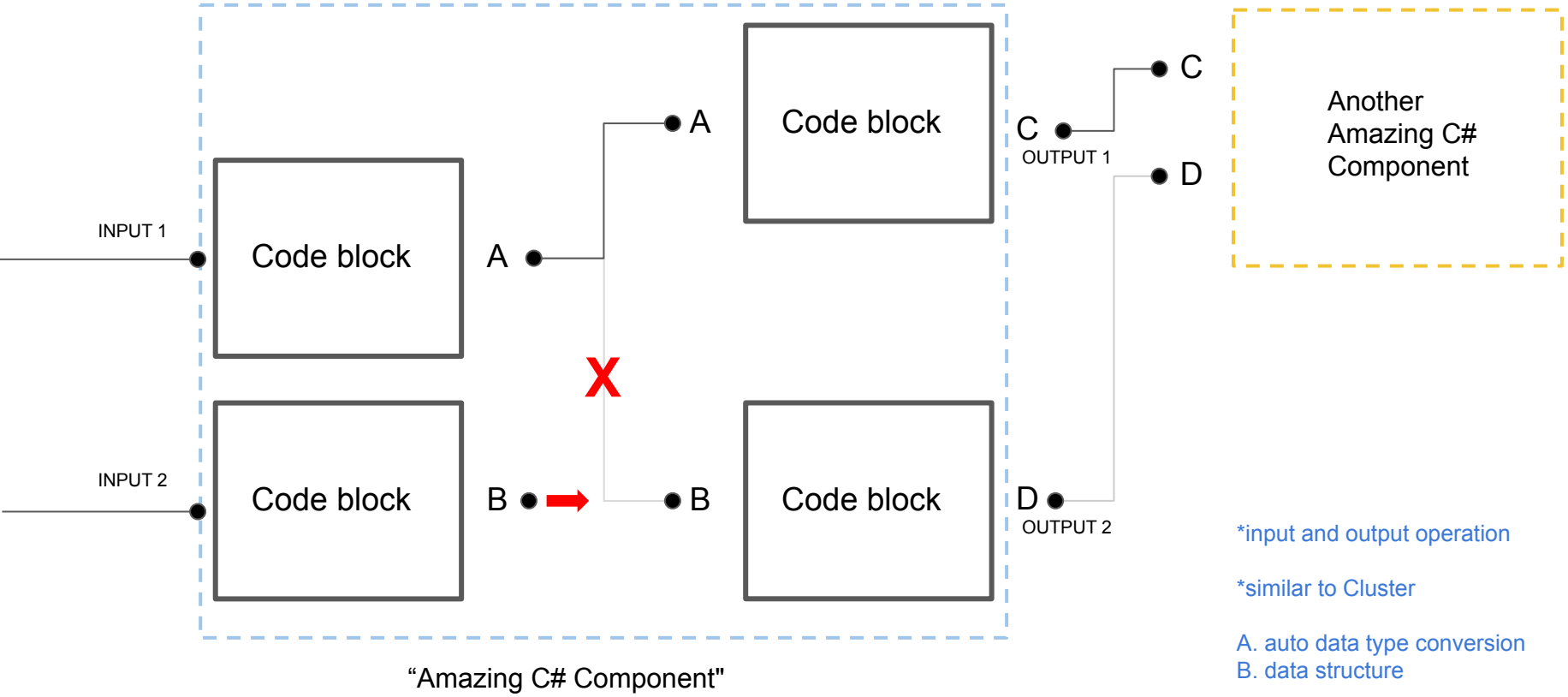
Rhino.NodeInCode namespace?
Serengeti (Rhino 7 WIP) Developer rhino grasshopper python
Nov '16 - Not sure if this is a dream or I missed some announcement. 😢 Is this what I think it is?

NodeInCode DivideCurve strange result
Grasshopper Developer
Apr '18 - ...hree strange results, am I doing something wrong? Canvas%20at%2017%3B02%3B35 private void RunScript(ref object A, ref object B, ref object C) { Rhino.**NodeInCode**.ComponentFunctionInfo cfi = Rhino.**NodeInCode**.Components.FindComponent("DivideCurve"); string[] warnings; object[] result = cfi.Evaluate(new object[]{...

Asynchronous issues with NodeInCode
Scripting
Apr '18 - Great Guilio! I have a question, what about cascading using **NodeInCode**? that is, create a definition (connected nodes) from code. Should we use asynchronous patterns? what considerations can you add for this? Thanks!

Rhinocommon NodeInCode doesn't work without starting grasshopper
Scripting
Dec '18 - Hey @piac i fiddled around with **nodeInCode** today and i cant recreate this: https://discourse.mcneel.com/t/using-the-**nodeincode**-namespace/49735/8 Using the **NodeInCode** namespace Do you need gras...

What is NodeInCode?



*input and output operation

*similar to Cluster

A. auto data type conversion
B. data structure

Create a custom function
using Rhinocommon default syntax

Curve.PointAt Method

Evaluates point at a curve parameter.

Namespace: Rhino.Geometry
Assembly: RhinoCommon (in RhinoCommon.dll)

Syntax

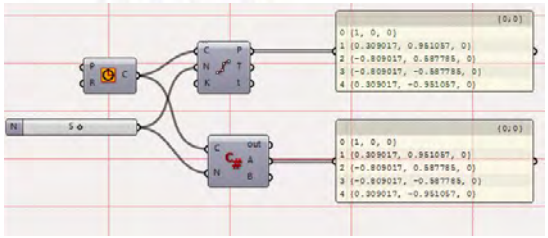
C# VB

```
public Point3d PointAt(
        double t
)
```

Parameters

t

Type: System.Double
Evaluation parameter.



Curve.DivideByCount Method (Int32, Boolean,Point3d[])

Divide the curve into a number of equal-length segments.

Namespace: Rhino.Geometry
Assembly: RhinoCommon (in RhinoCommon.dll)

Syntax

C# VB

```
public double[] DivideByCount(
        int segmentCount,
        bool includeEnds,
        out Point3d[] points
)
```

Parameters

segmentCount
Type: System.Int32
Segment count. Note that the number of division points may differ from the segment count.

includeEnds
Type: System.Boolean
If true, then the point at the start of the first division segment is returned.

points
Type:Rhino.Geometry.Point3d[]
A list of division points. If the function returns successfully, this point-array will be filled in.

Return Value

Type:Double[]
Array containing division curve parameters on success, null on failure.

Create a custom function using NodeInCode

Method 1

Components.FindComponent Method

Finds a component given its full name.

Namespace: Rhino.NodeInCode
Assembly: RhinoCommon (in RhinoCommon.dll)

Syntax

C# VB

```
public static ComponentFunctionInfo FindComponent(
        string fullName
)
```

Parameters

fullName
Type: System.String
The name, including its library name and a period if it is made by a third-party.

Return Value
Type: ComponentFunctionInfo
[Missing <returns> documentation for "M:Rhino.NodeInCode.Components.FindComponent(System.String)"]

1

ComponentFunctionInfo.Evaluate Method

Evaluates the component with a set of arguments. There needs to be an argument for each input param, and each output param gives an entry in the output array.

Namespace: Rhino.NodeInCode
Assembly: RhinoCommon (in RhinoCommon.dll)

Syntax

C# VB

```
public abstract Object[] Evaluate(
        IEnumerable args,
        bool keepTree,
        out string[] warnings
)
```

Parameters

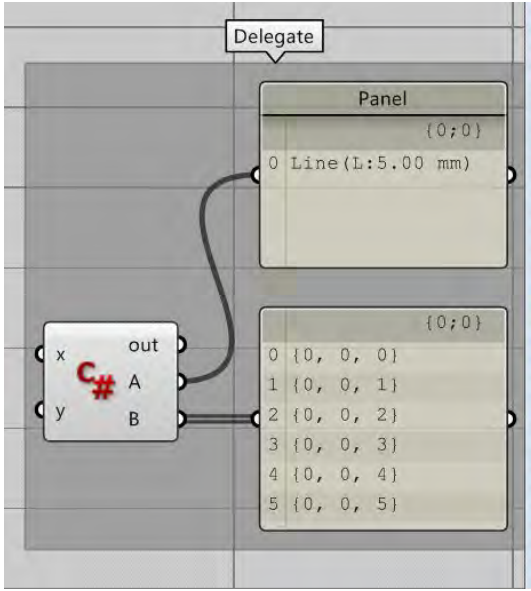
args
Type: System.Collections.IEnumerable
The arguments list. Each item is assigned to each input param, in order.
keepTree
Type: System.Boolean
A value indicating whether trees should be considered valid inputs, and should be returned. In this case, output variables are not simplified to common types.
warnings
Type:System.String[]
A possible list of warnings, or null.

Return Value
Type:Object[]
An array of objects, each representing an output result.

2

Create a custom function using NodeInCode

Method 2




```
using Rhino.NodeInCode;

/// <summary>
/// This class will be instantiated on demand by the Script component.
/// </summary>
public class Script_Instance : GH_ScriptInstance
{
Utility functions

Members

  /**/
  private void RunScript(object x, object y, ref object A, ref object B)
  {

    Func<object,object,object,object> com1 = (Func<object,object,object,object>) Components.FindComponent("LineSDL").Delegate;
    object ot1 = com1(Point3d.Origin, Vector3d.ZAxis, 5);

    //A = com1.DynamicInvoke(Point3d.Origin, Vector3d.ZAxis, 5);

    A = ot1;


    Func<object,object,object,object> com2 = (Func<object,object,object,object>) Components.FindComponent("DivideCurve").Delegate;
    //IList<object> ot2 = com2(ot1, 5, false) as IList<object>;

    IList<object> ot2 = com2.DynamicInvoke(ot1, 5, false) as IList<object>;

    B = ot2[0];
```

ComponentFunctionInfo.Delegate Property

Returns a delegate that can be directly invoked using a list of arguments. This flattens trees.

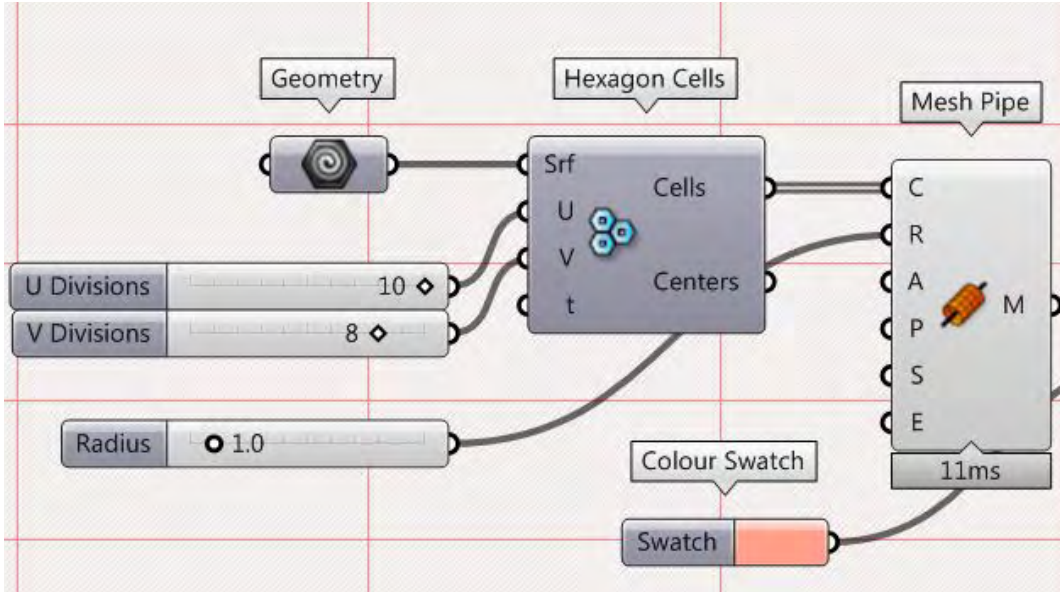
Namespace: Rhino.NodeInCode
Assembly: RhinoCommon (in RhinoCommon.dll)

Syntax

C# VB

```
public Delegate Delegate { get; }
```

Step1. Name Inspection




```
namespace Attribute
{
    public class AttributeComponent : GH_Component
    {
        /// <summary>
        /// Each implementation of GH_Component mus
        /// constructor without any arguments.
        /// Category represents the Tab in which th
        /// Subcategory the panel. If you use non-e
        /// new tabs/panels will automatically be c
        /// </summary>
        public AttributeComponent()
          : base("Attribute", "Nickname",
              "Description",
              "Category", "Subcategory")
        {
        }
```

NodeInCodeTable.GetDynamicMemberNames Method

Returns all additional names in the table.

Namespace: Rhino.NodeInCode
Assembly: RhinoCommon (in RhinoCommon.dll)

Syntax

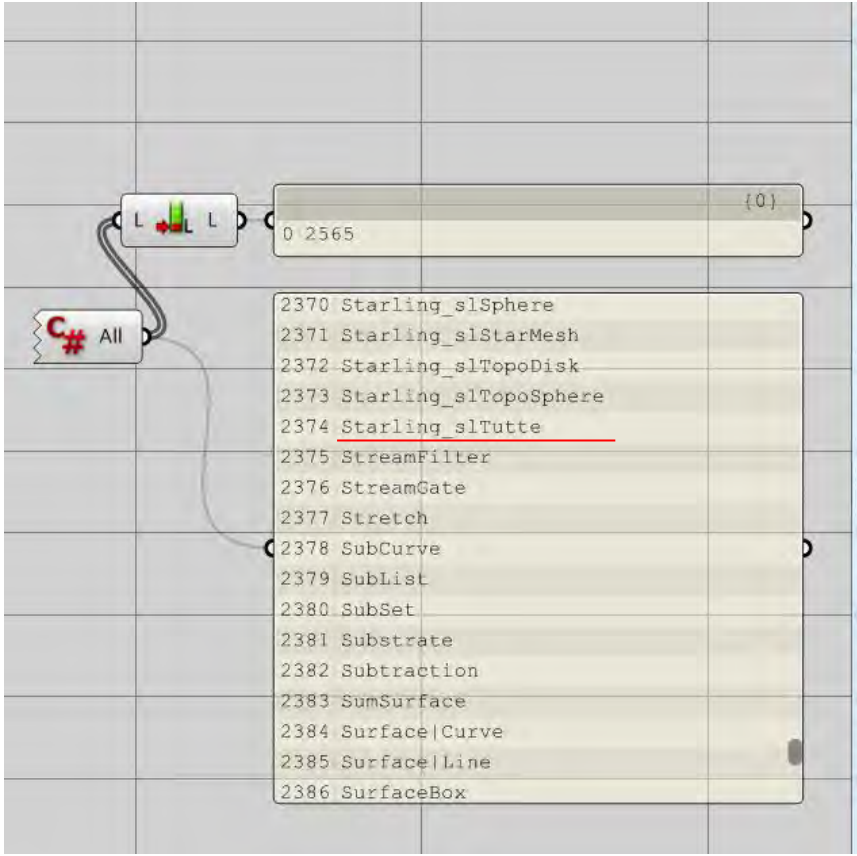
C# VB

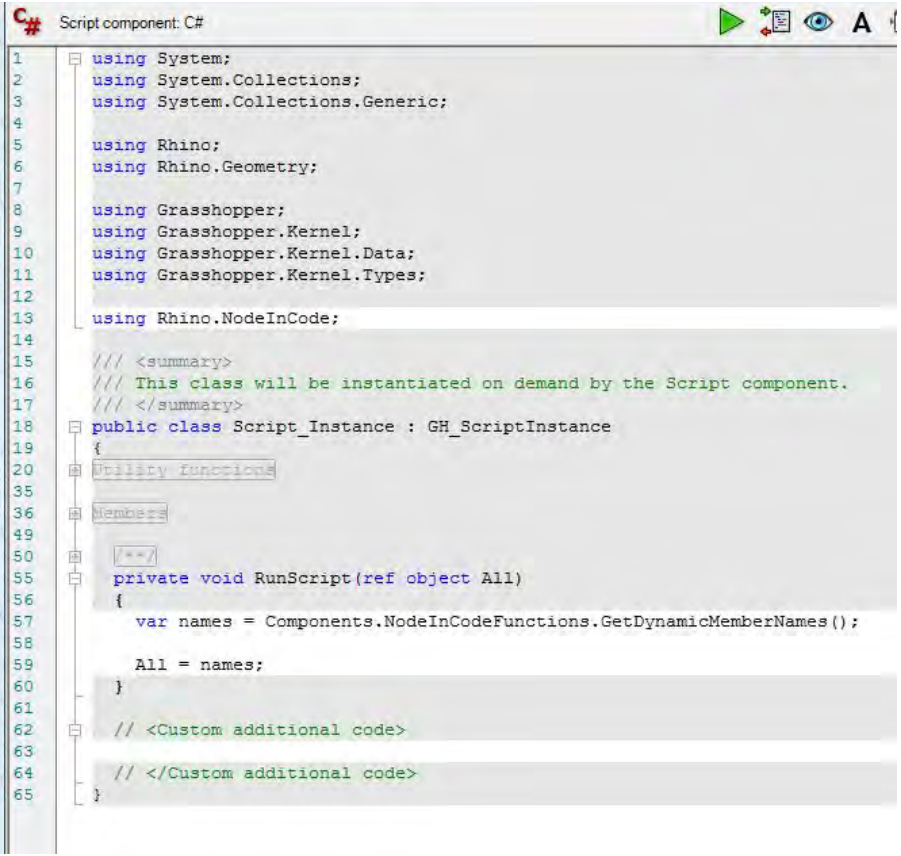
```
public override IEnumerable<string> GetDynamicMemberNames()
```

Return Value

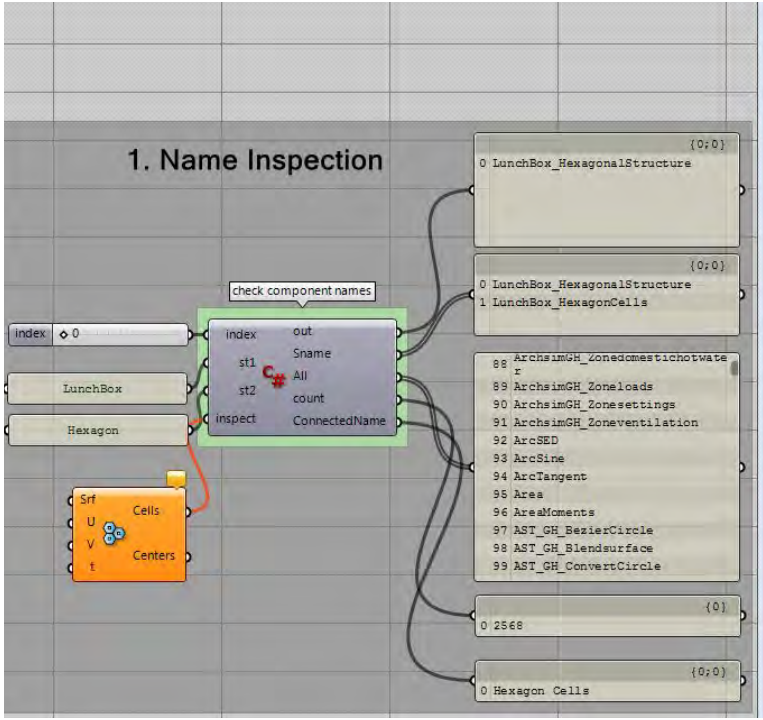
Type: IEnumerable<String>
[Missing <returns> documentation for "M:Rhino.NodeInCode.NodeInCodeTable.GetDynamicMemberNames"]

Search for available component names





Use name search / filter and reflection




```
using System.Linq;
using Rhino.NodeInCode;

/// <summary>
/// This class will be instantiated on demand by the Script component.
/// </summary>
public class Script_Instance : GH_ScriptInstance
{
#region Utility functions

#region Members

  /**/
  private void RunScript(int index, string st1, string st2, object inspect, ref object Sname, ref object All, ref
  {
    var names = Rhino.NodeInCode.Components.NodeInCodeFunctions.GetDynamicMemberNames();

    var iname = names.Where(n => n.ToLower().Contains(st1.ToLower()) && n.ToLower().Contains(st2.ToLower()));

    List<string> connected = new List<string>();

    Component.Params.Input[3].Sources.ToList().ForEach(n =>
      {
      connected.Add(n.Attributes.GetTopLevel.DocObject.Name);
      });


    if(iname.Count() > index){
      Print(iname.ElementAt(index));
    }else{
      Print("set smaller index");
    }

    Sname = iname;
    All = names;
    count = names.Count();
    ConnectedName = connected;

  }
```

ComponentFunctionInfo.Evaluate Method

Evaluates the component with a set of arguments. There needs to be an argument for each input param, and each output param gives an entry in the output array.

Namespace: Rhino.NodeInCode
Assembly: RhinoCommon (in RhinoCommon.dll)

Syntax

C# VB

```
public abstract Object[] Evaluate(
        IEnumerable args,
        bool keepTree,
        out string[] warnings
)
```

Parameters

args
Type: System.Collections.IEnumerable
The arguments list. Each item is assigned to each input param, in order.

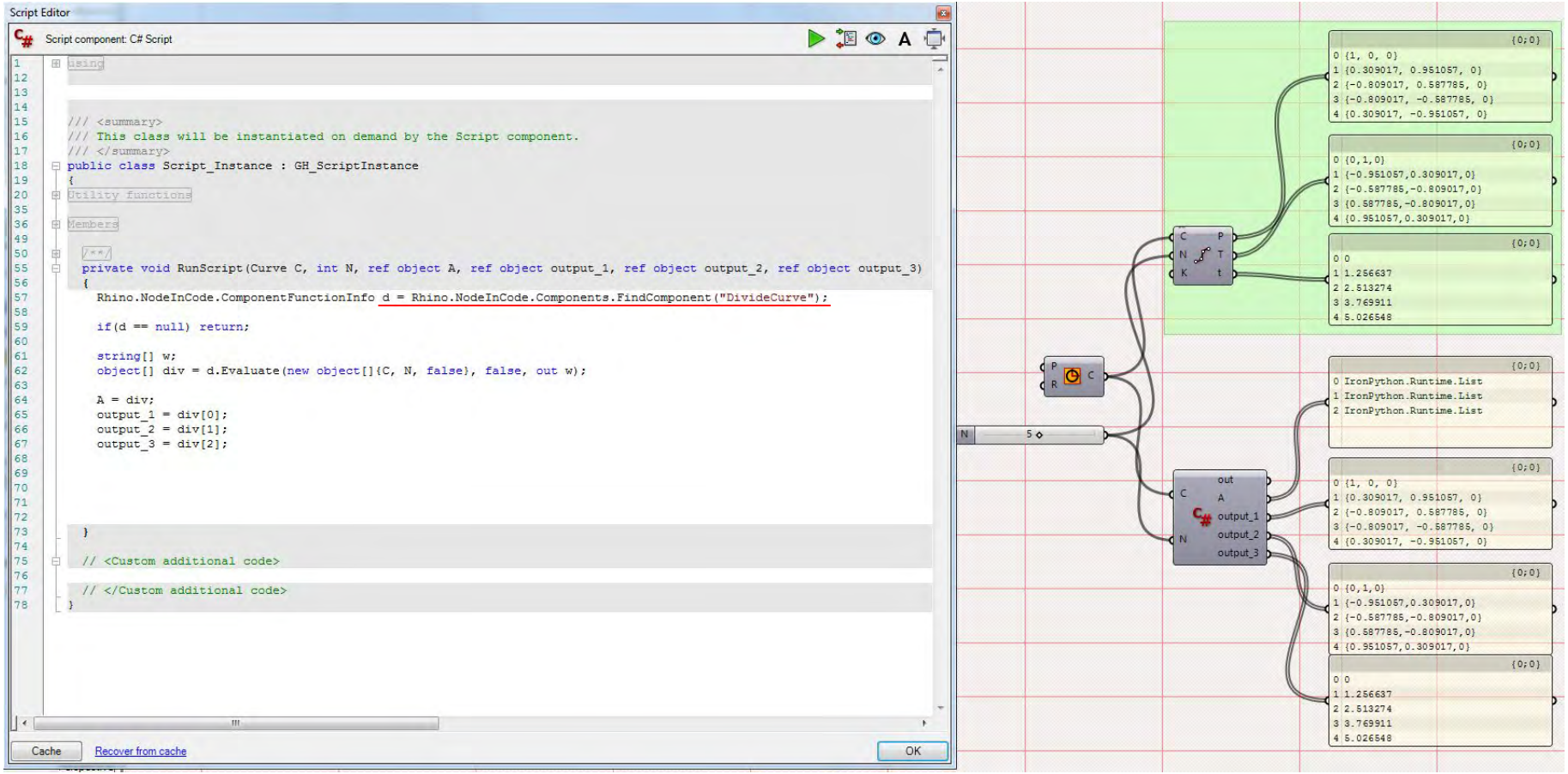
keepTree
Type: System.Boolean
A value indicating whether trees should be considered valid inputs, and should be returned. In this case, output variables are not simplified to common types.

warnings
Type:System.String[]
A possible list of warnings, or null.

Return Value

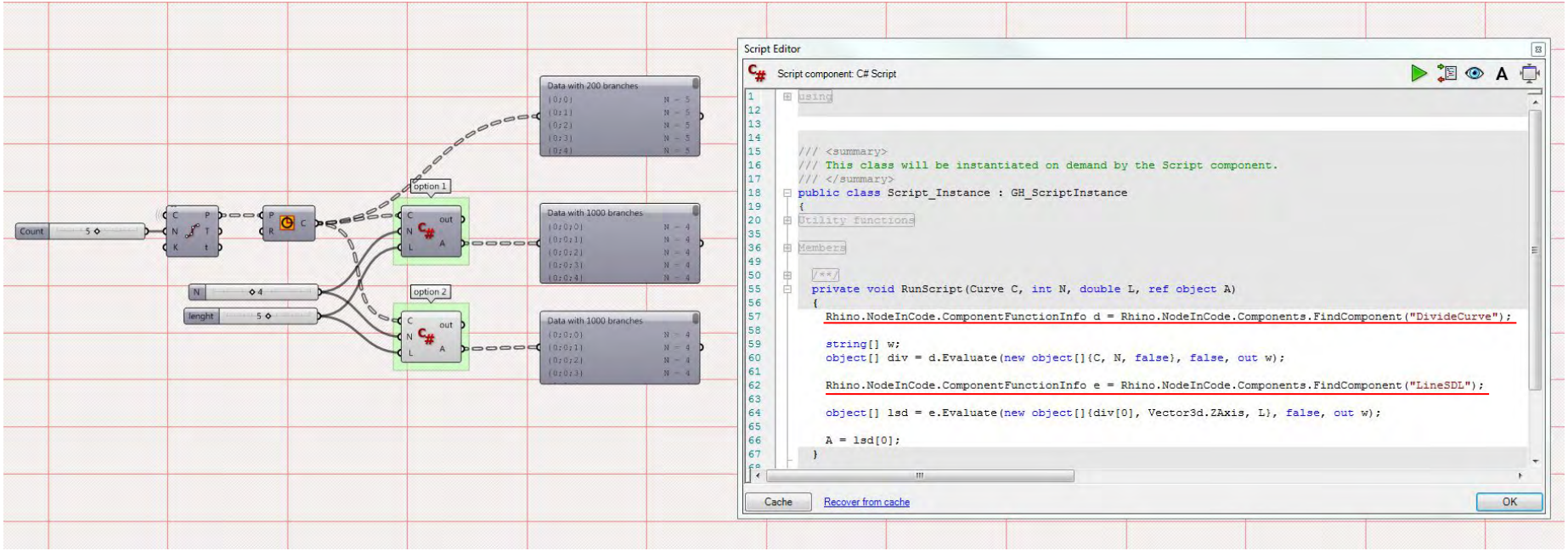
Type:Object[]
An array of objects, each representing an output result.

Step2. NodeInCode syntax



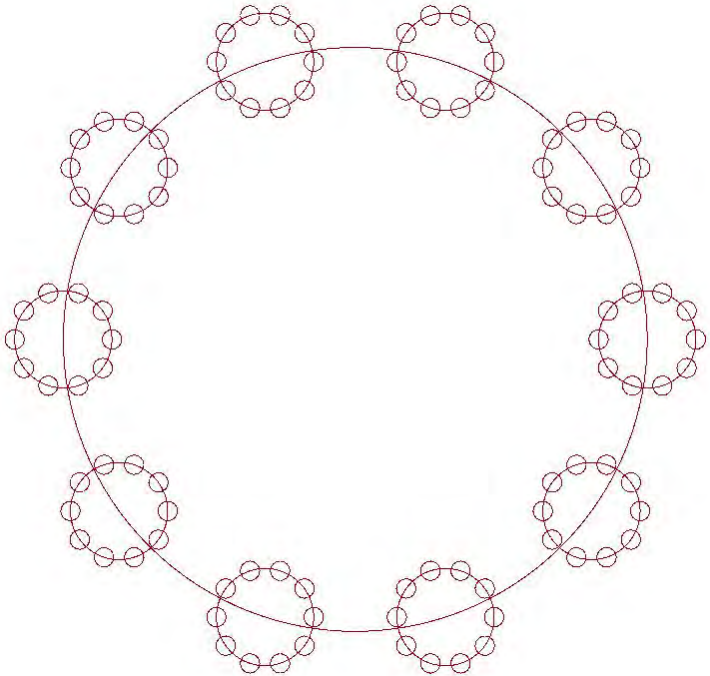

```
using

/// <summary>
/// This class will be instantiated on demand by the Script component.
/// </summary>
public class Script_Instance : GH_ScriptInstance
{
Utility functions

Members

  /**/
  private void RunScript(Curve C, int N, ref object A, ref object output_1, ref object output_2, ref object output_3)
  {
    Rhino.NodeInCode.ComponentFunctionInfo d = Rhino.NodeInCode.Components.FindComponent("DivideCurve");

    if(d == null) return;

    string[] w;
    object[] div = d.Evaluate(new object[]{C, N, false}, false, out w);

    A = div;
    output_1 = div[0];
    output_2 = div[1];
    output_3 = div[2];




  }

  // <Custom additional code>

  // </Custom additional code>
}
```

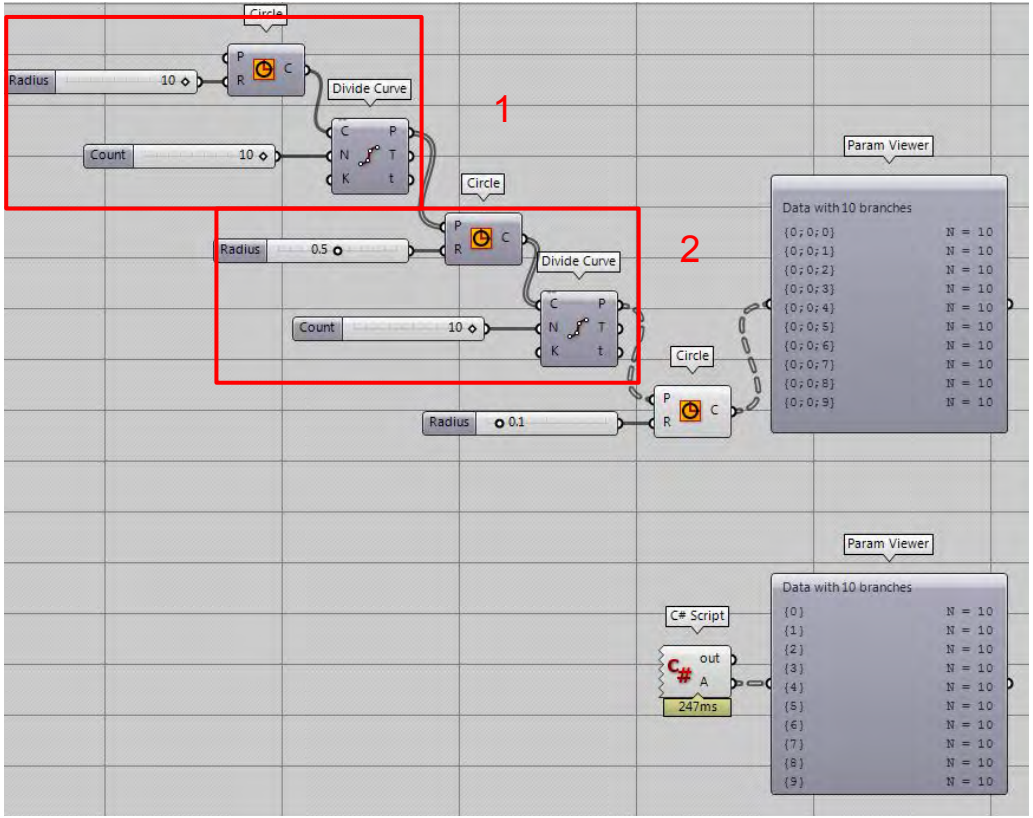
NodeInCode - multiple components



You can chain up a series of components together without considering much about data structure

NodeInCode - multiple components





NodeInCode - components without documentation

Grasshopper API - Search

Voronoi

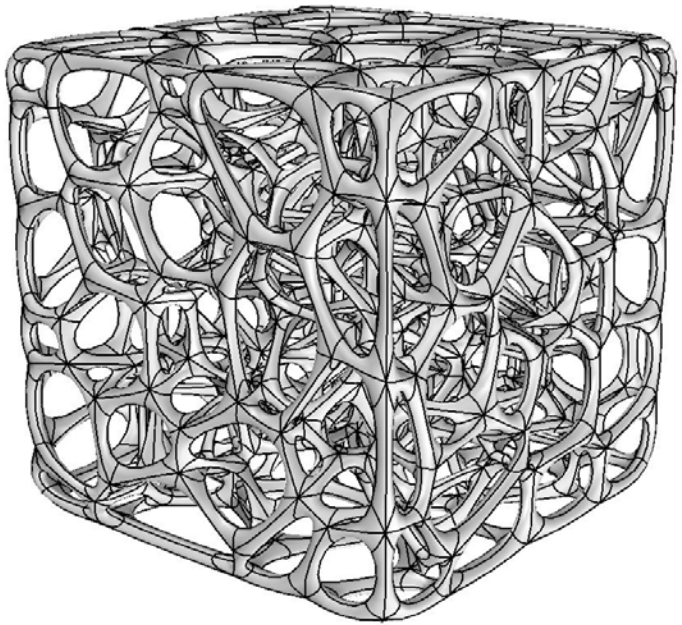
Sort by title

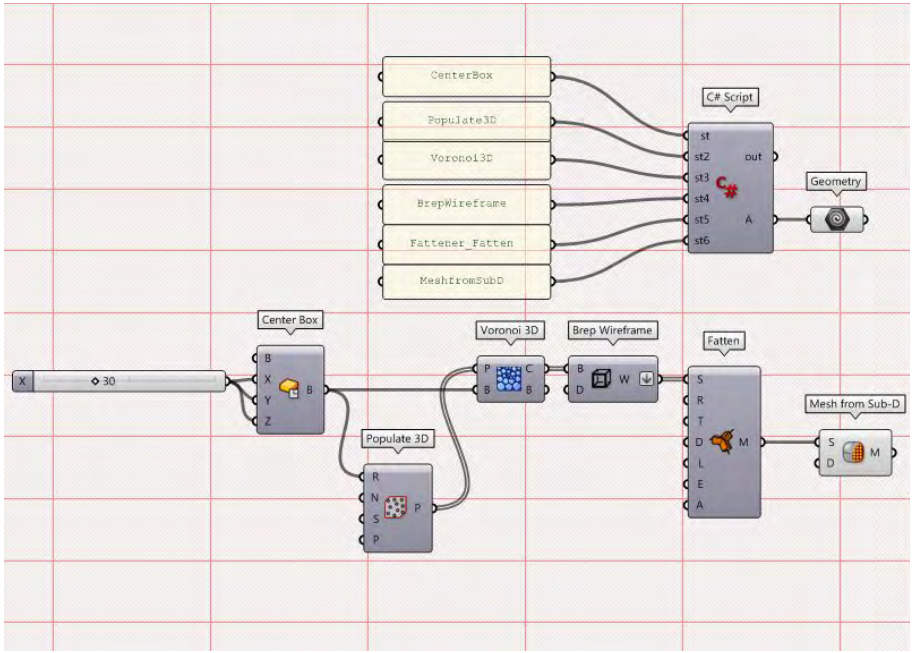
1. GH_Canvas.DisplayVoronoiWarning Property
2. GH_Canvas Properties
3. GH_Canvas Class

- Grasshopper.Kernel.Geometry.Voronoi
  - Cell2
  - Cell3
  - Cell3Facet
  - Solver

```
using Rhino;
using Rhino.Collections;
using Rhino.Geometry;
using Rhino.Runtime;
using System;
using System.Collections.Generic;
using System.Linq;

namespace Grasshopper.Kernel.Geometry.Voronoi
{
  public class Cell3
  {
    private List<Cell3Facet> m_facets;
    private Point3d m_center;
    private double m_tolerance;

    public Cell3(Point3d center, Box box)
    {
      this.m_facets = new List<Cell3Facet>();
      if (((Box) ref box).get_Volume() == 0.0)
        throw new InvalidOperationException("Cell box must have a positive volume");
      this.m_center = center;
      Point3d[] corners = ((Box) ref box).GetCorners();
      this.m_facets.Add(new Cell3Facet((IEnumerable<Point3d>) new Point3d[5]
      {
        corners[3],
        corners[2],
        corners[1],
        corners[0],
        corners[3]
      }));
      this.m_facets.Add(new Cell3Facet((IEnumerable<Point3d>) new Point3d[5]
      {
        corners[0],
        corners[1],
        corners[5],
        corners[4],
        corners[0]
      }));
      this.m_facets.Add(new Cell3Facet((IEnumerable<Point3d>) new Point3d[5]
      {
```

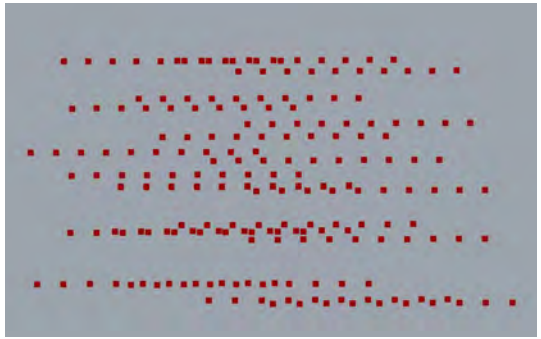
Example - documentation is not available /
3rd party function (Rhino7 WIP - using SubD component)

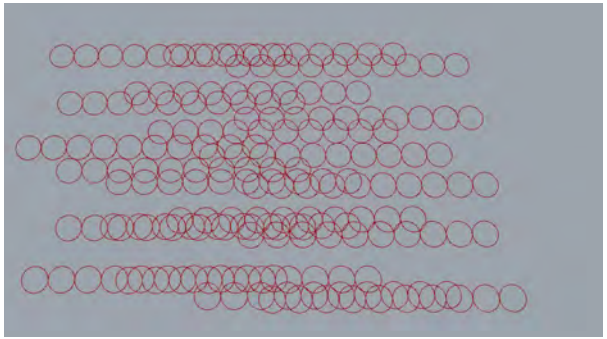


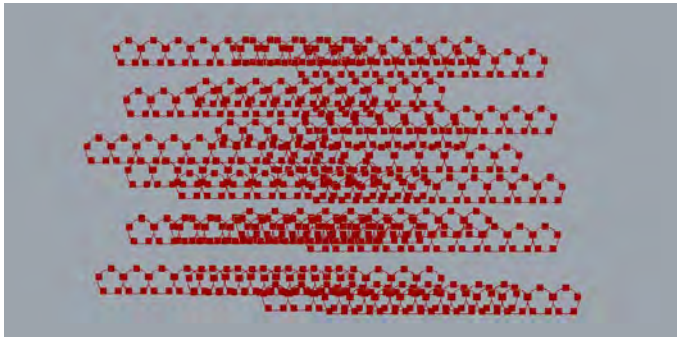


NodeInCode - Data Flow Control

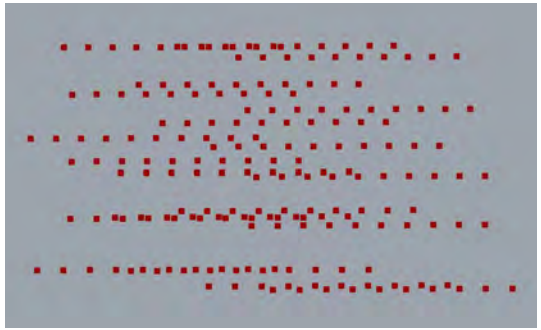
One to one / one to multiple (multiple to one)

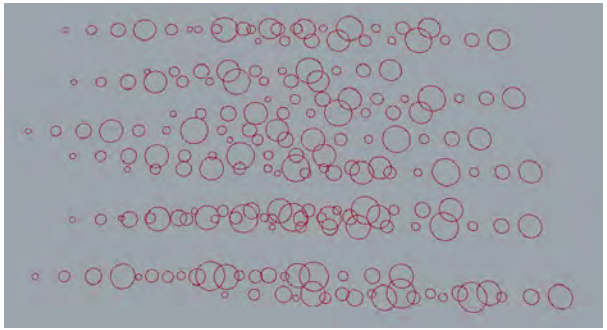


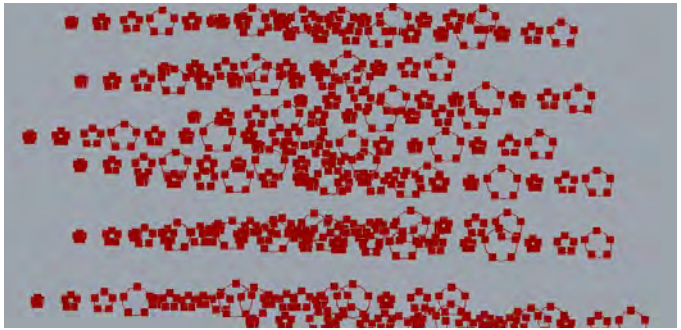




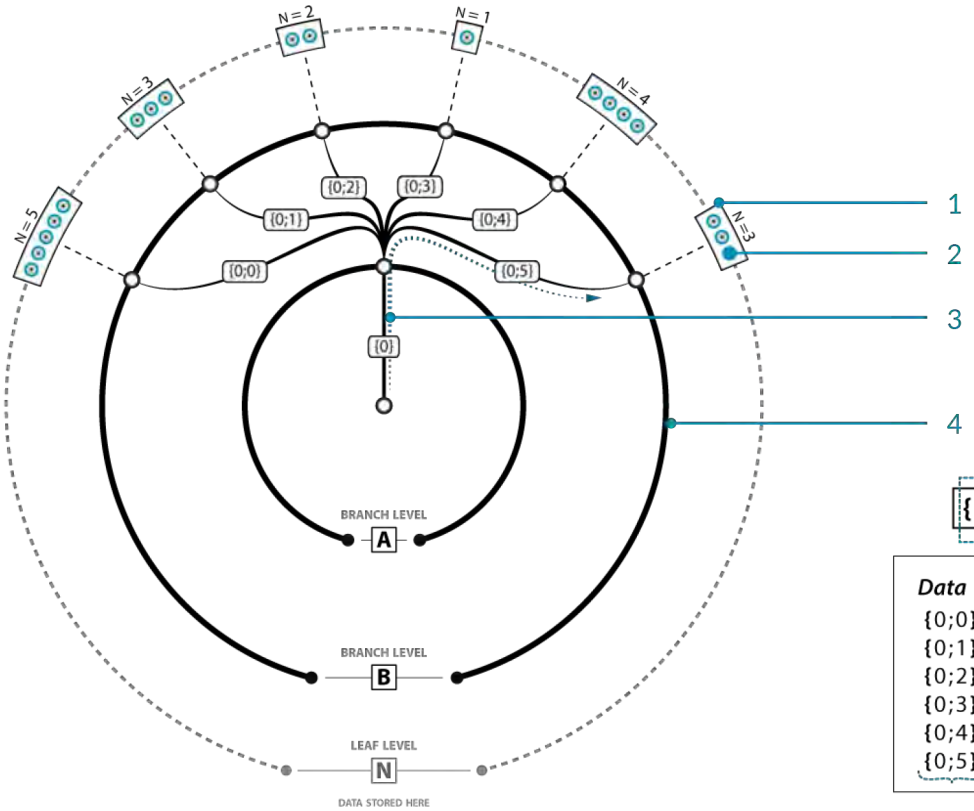
multiple to multiple







Data Structure in GH



1
2
3
4

{A;B} (i)

Data with 6 branches	
{0;0}	N = 5
{0;1}	N = 3
{0;2}	N = 2
{0;3}	N = 1
{0;4}	N = 4
{0;5}	N = 3

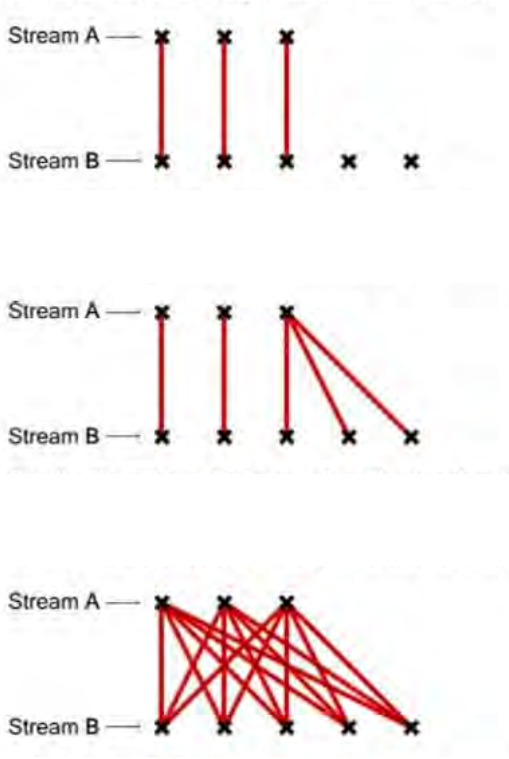
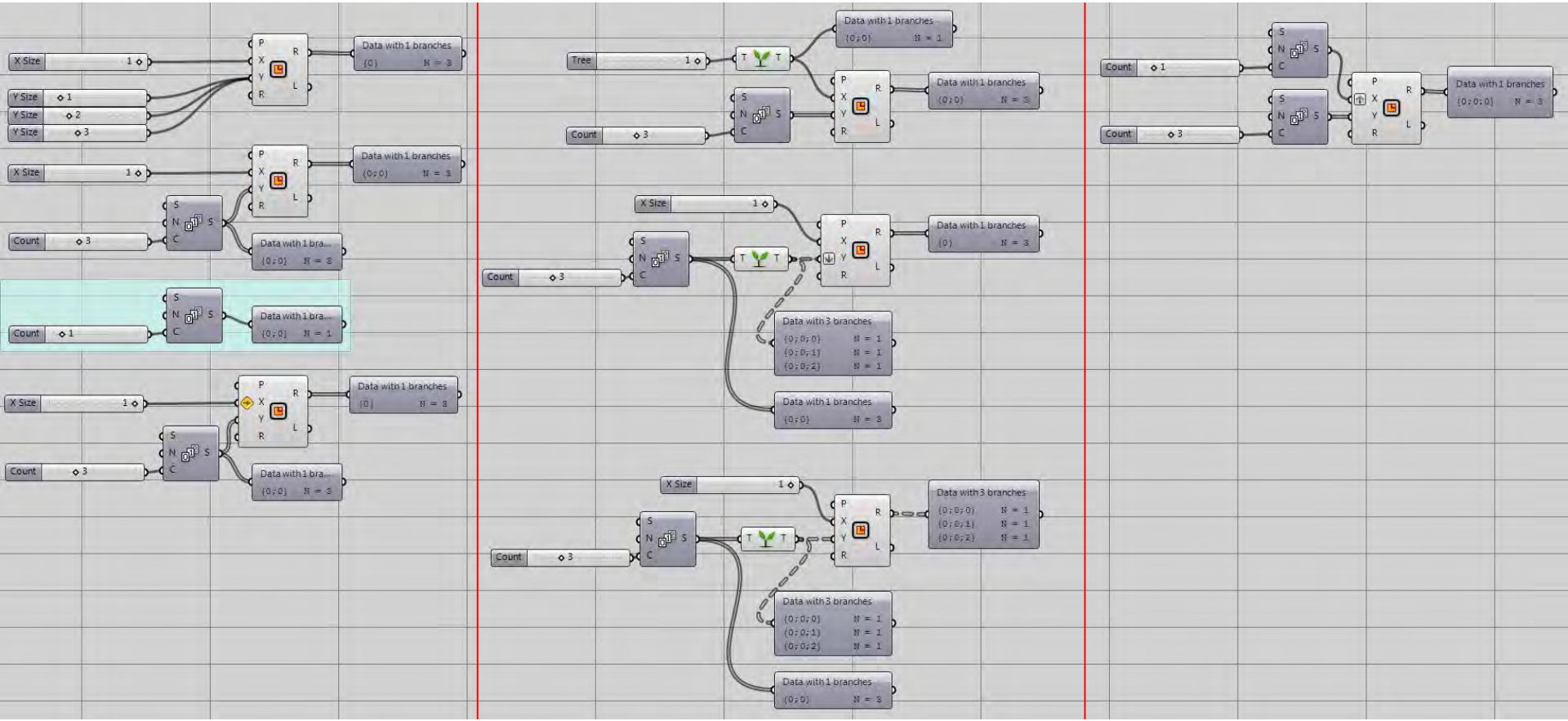
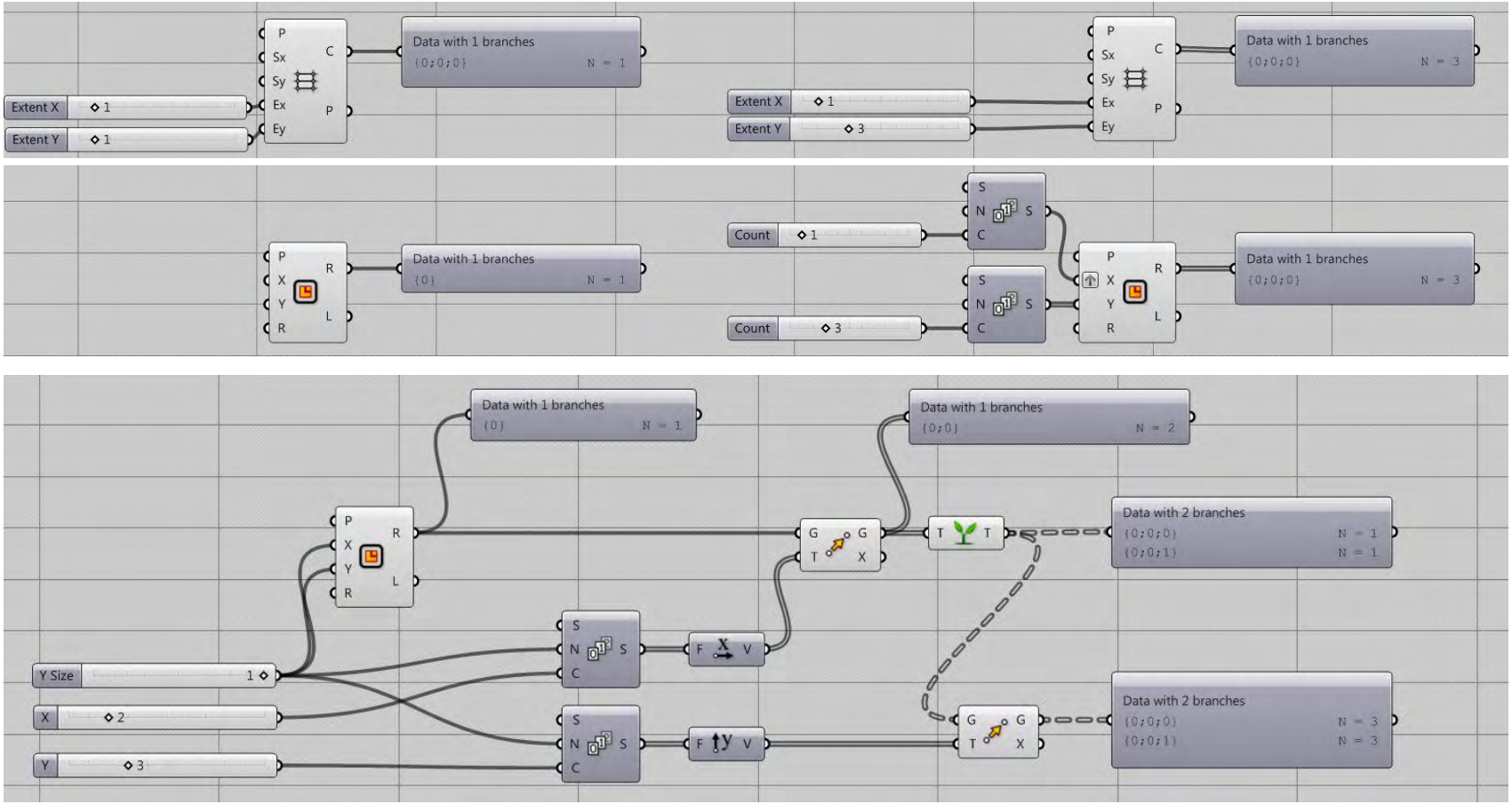
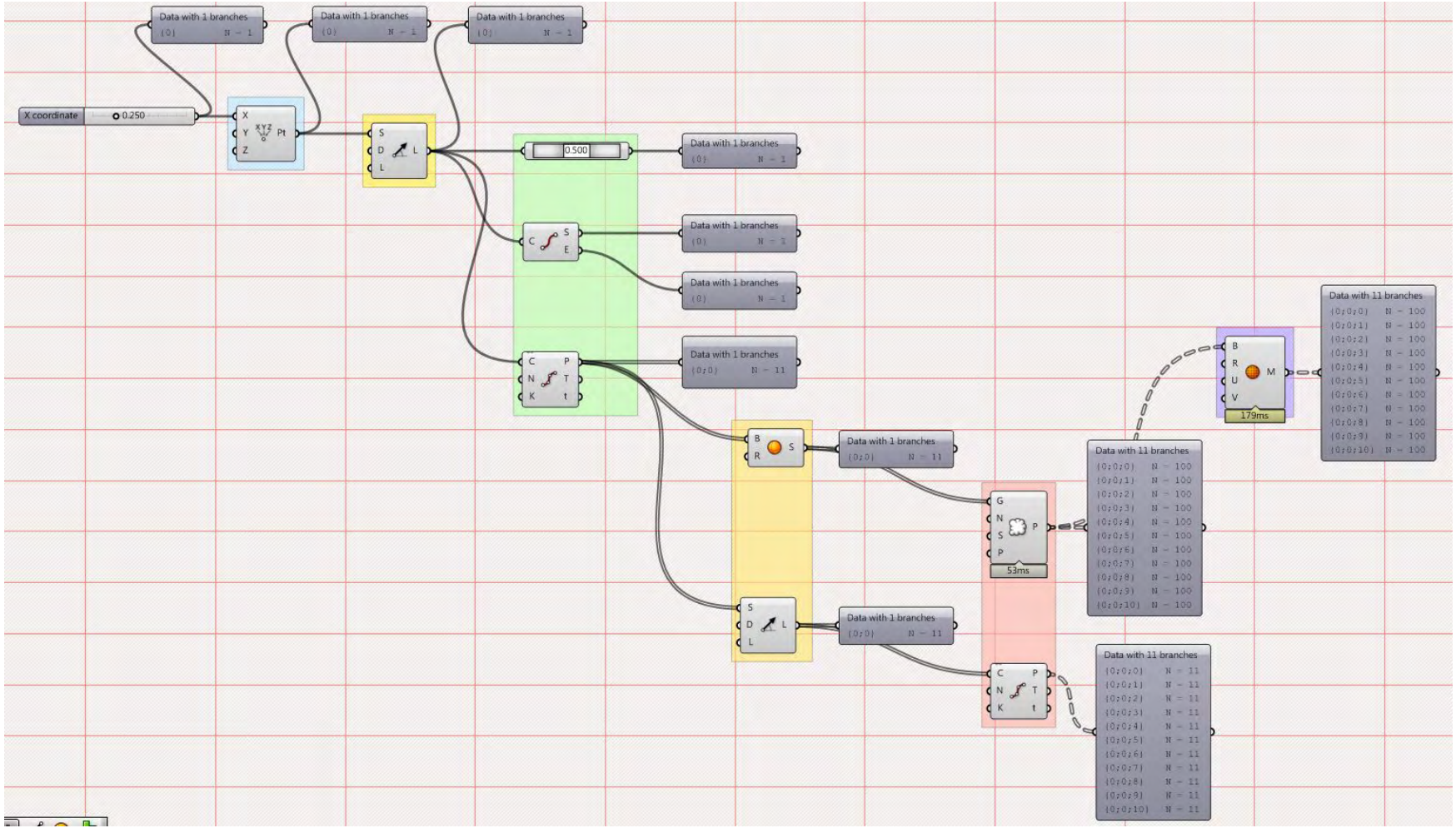


image source: Google


X Size
1
Y Size
1
Y Size
2
Y Size
3
P
X
Y
R
R
L
Data with 1 branches
{0} N = 3
X Size
1
Count
3
S
N
C
S
Data with 1 bra...
{0;0} N = 3
Count
1
Data with 1 bra...
{0;0} N = 1
X Size
1
Count
3
Data with 1 branches
{0} N = 3
Data with 1 bra...
{0;0} N = 3
Tree
1
T
Data with 1 branches
{0;0} N = 1
Count
3
Data with 1 branches
{0;0} N = 3
X Size
1
Count
3
Data with 1 branches
{0} N = 3
Data with 3 branches
{0;0;0} N = 1
{0;0;1} N = 1
{0;0;2} N = 1
Data with 1 branches
{0;0} N = 3
X Size
1
Count
3
Data with 3 branches
{0;0;0} N = 1
{0;0;1} N = 1
{0;0;2} N = 1
Data with 3 branches
{0;0;0} N = 1
{0;0;1} N = 1
{0;0;2} N = 1
Data with 1 branches
{0;0} N = 3
Count
1
Count
3
Data with 1 branches
{0;0;0} N = 3

Data Structure




Data with 1 branches
Data with 1 branches
Data with 1 branches
X coordinate
0.250
X
Y
Z
Pt
S
D
L
0.500
C
S
E
C
N
K
P
T
t
Data with 1 branches
Data with 1 branches
Data with 1 branches
Data with 1 branches
B
R
S
Data with 1 branches
S
D
L
Data with 1 branches
G
N
S
P
P
53ms
Data with 11 branches
C
N
K
P
T
t
Data with 11 branches
B
R
U
V
M
179ms
Data with 11 branches

Create new storage

Curve.DivideByCount Method (Int32, Boolean,Point3d[])

Divide the curve into a number of equal-length segments.

Namespace: Rhino.Geometry
Assembly: RhinoCommon (in RhinoCommon.dll)

Syntax

C# VB

```
public double[] DivideByCount(
        int segmentCount,
        bool includeEnds,
        out Point3d[] points
)
```

Derived from Input geometry

Curve.PointAt Method

Evaluates point at a curve parameter.

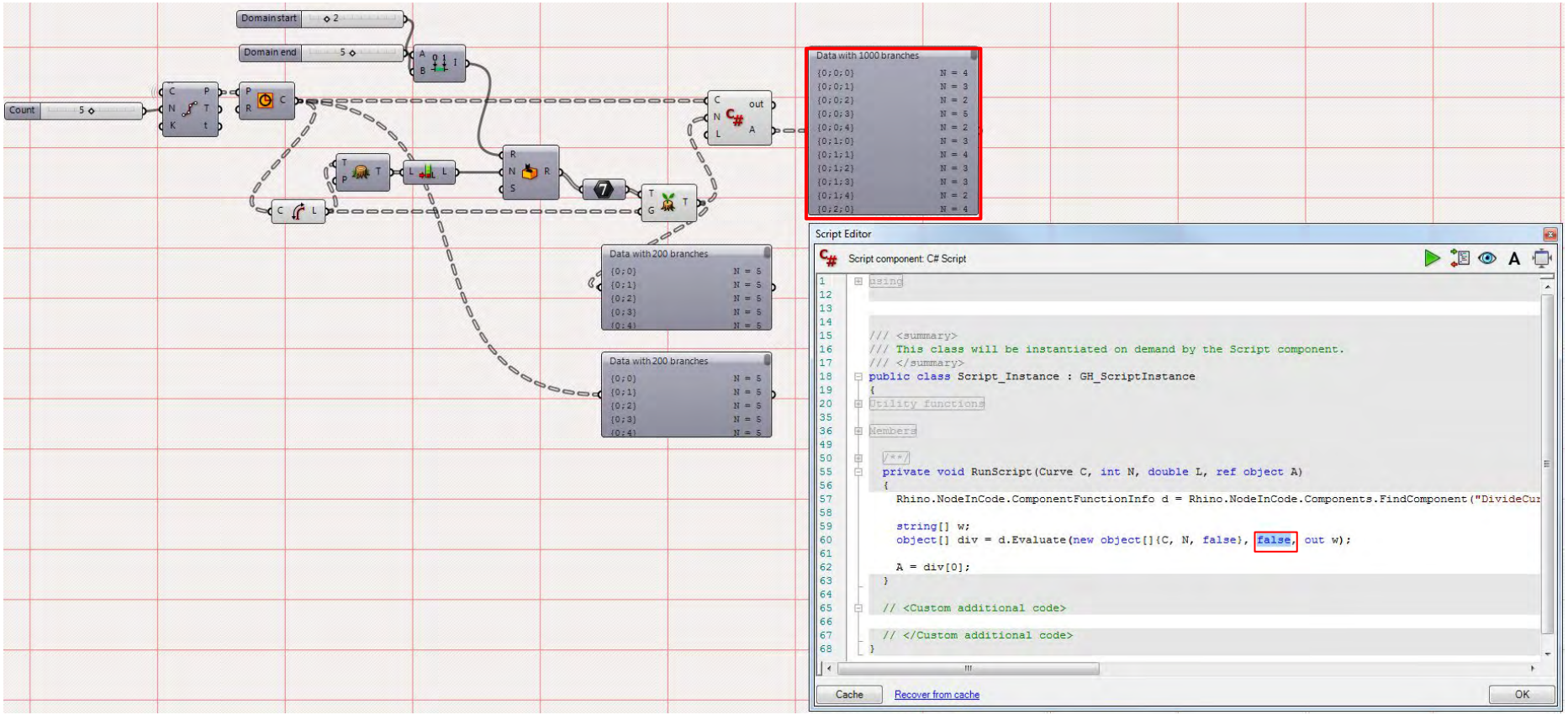
Namespace: Rhino.Geometry
Assembly: RhinoCommon (in RhinoCommon.dll)

Syntax

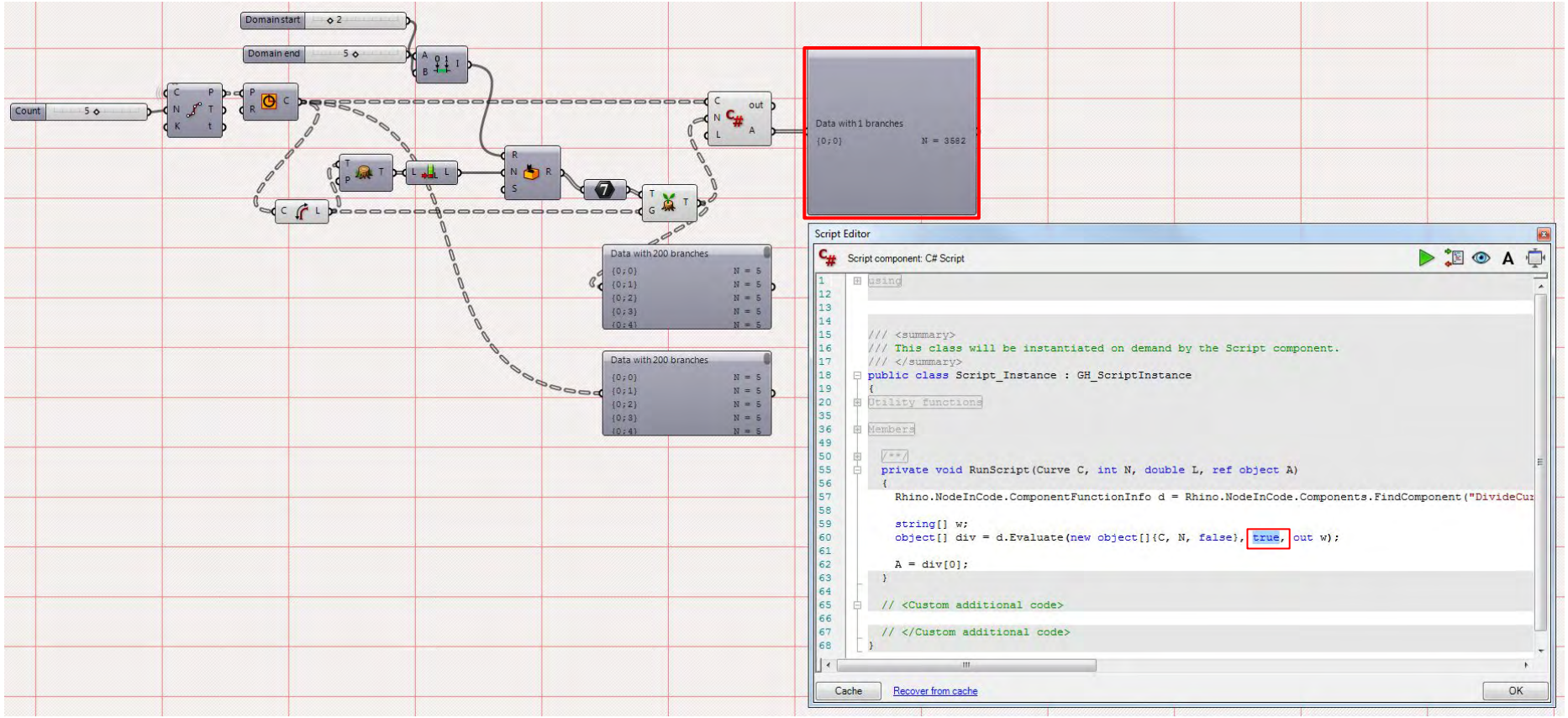
C# VB

```
public Point3d PointAt(
        double t
)
```

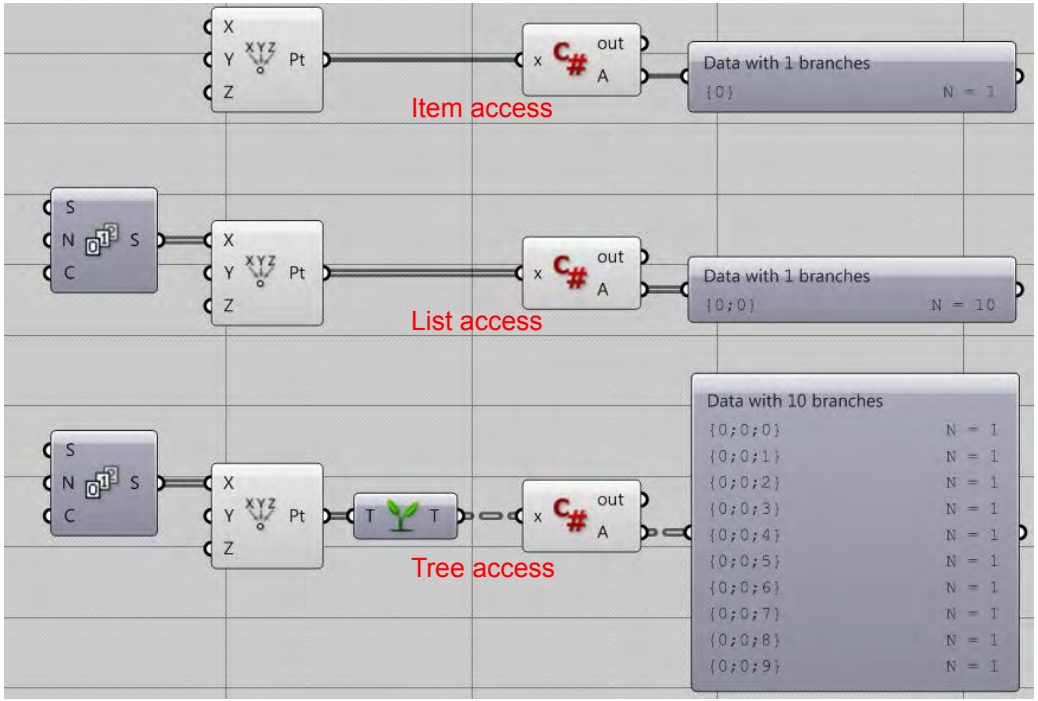
Data Structure Control with NodeInCode



Data Structure Control with NodeInCode



Data Structure - data access

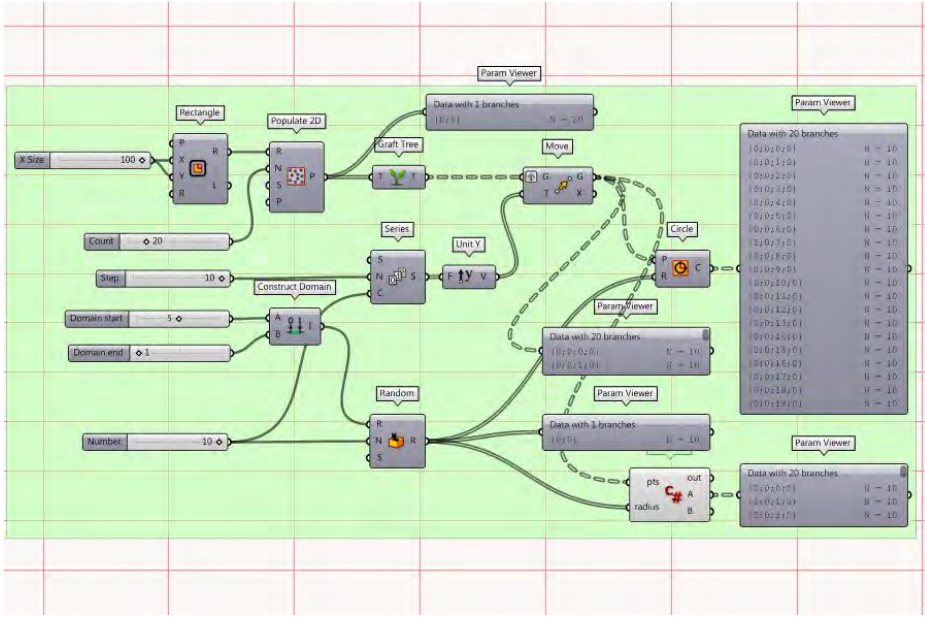



```
private void RunScript(Point3d x, ref object A)
{
  A = new Circle(x, 5);
}
```

```
private void RunScript(List<Point3d> x, ref object A)
{
  var c = new List<Circle>();

  foreach ( var pt in x){

    Circle cir = new Circle(pt, 5);
    c.Add(cir);
  }
  A = c;
}
```

```
private void RunScript(DataTree<Point3d> x, ref object A)
{
  DataTree<Circle> c = new DataTree<Circle>();

  foreach ( var p in x.Paths){
    foreach ( var pt in x.Branch(p)){

      Circle cir = new Circle(pt, 5);
      c.Add(cir, p);
    }
  }
  A = c;
}
```

Data matching with multiple objects
Tree / List Access solution

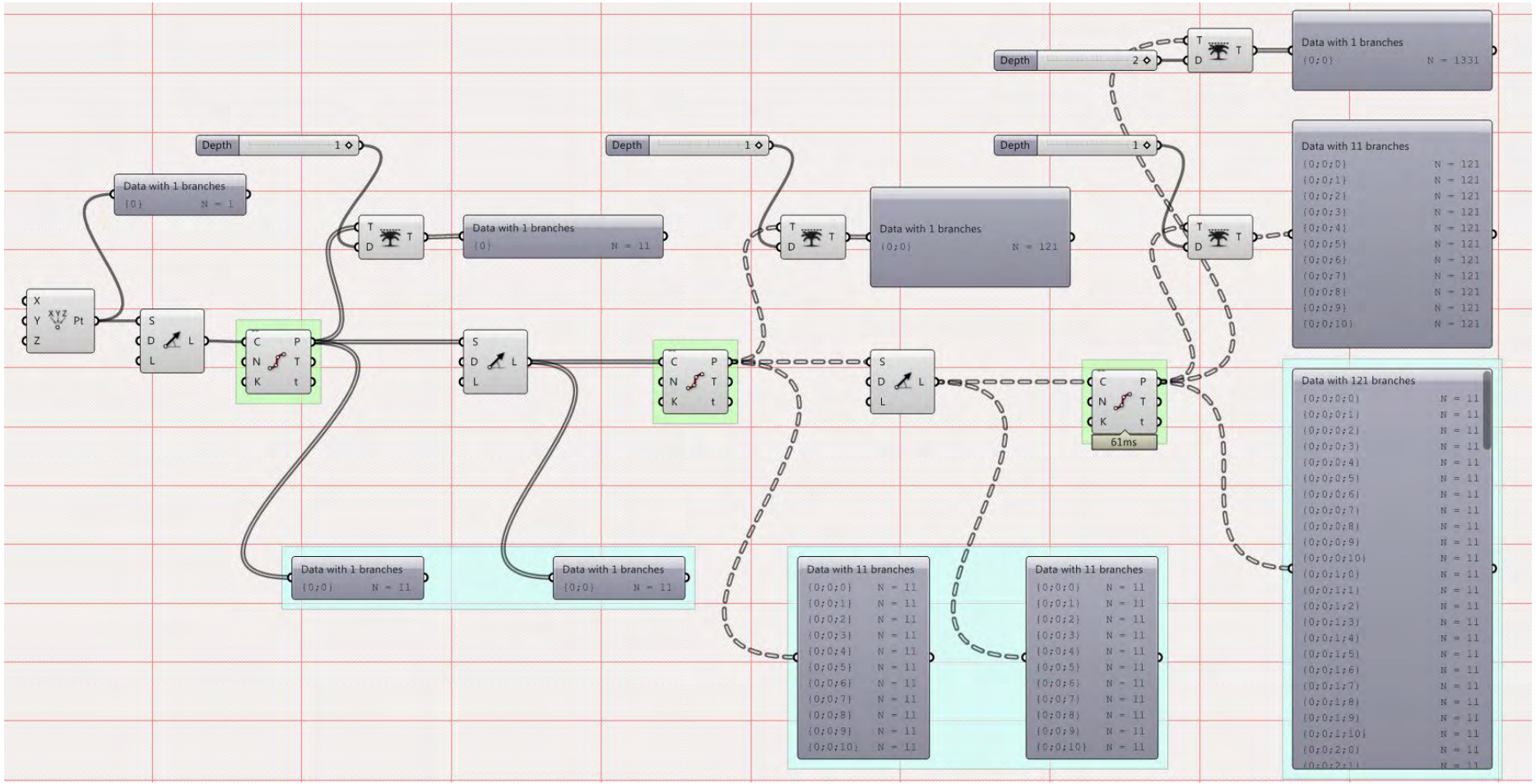


Script Editor

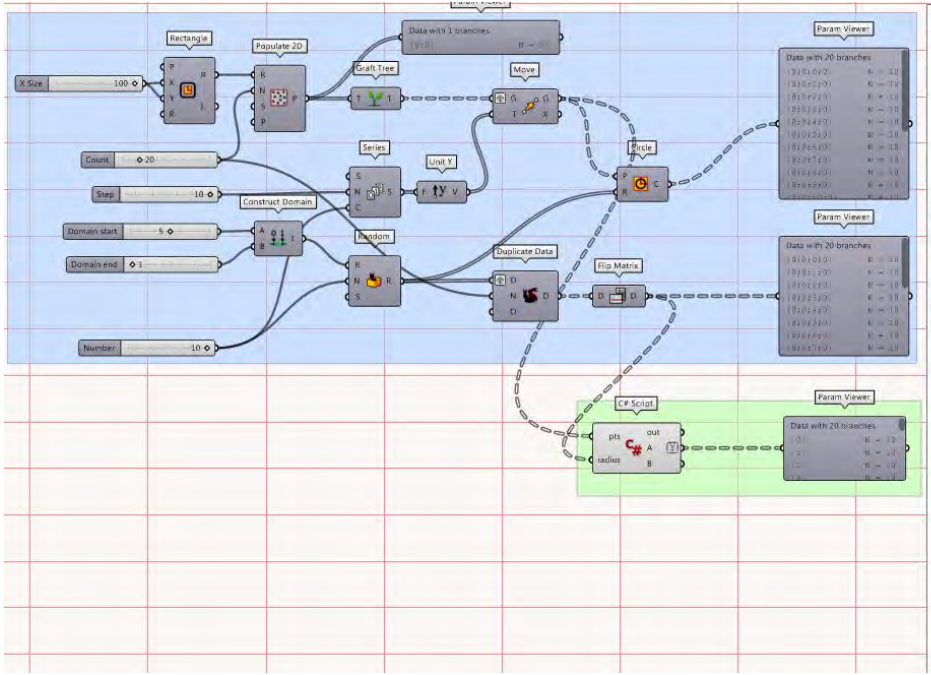
Script component: C# Script

```
using

/// <summary>
/// This class will be instantiated on demand by the Script component.
/// </summary>
public class Script_Instance : GH_ScriptInstance
{
Utility functions

Members

  /*...*/
  private void RunScript(DataTree<Point3d> pts, List<double> radius, ref object A, ref object B)
  {
    string[] error;

    Rhino.NodeInCode.ComponentFunctionInfo d = Rhino.NodeInCode.Components.FindComponent("Circle");
    object[] drawCircle = d.Evaluate(new object[]{pts, radius}, true, out error);

    A = drawCircle[0];

  }

  // <Custom additional code>

  // </Custom additional code>
}
```

Cache Recover from cache OK


Depth 1
Data with 1 branches {0} N = 1
Pt
Data with 1 branches {0} N = 11
Depth 1
Data with 1 branches {0;0} N = 121
Depth 2
Data with 1 branches {0;0} N = 1331
Depth 1
Data with 11 branches
{0;0;0} N = 121
{0;0;1} N = 121
{0;0;2} N = 121
{0;0;3} N = 121
{0;0;4} N = 121
{0;0;5} N = 121
{0;0;6} N = 121
{0;0;7} N = 121
{0;0;8} N = 121
{0;0;9} N = 121
{0;0;10} N = 121
61ms
Data with 1 branches {0;0} N = 11
Data with 1 branches {0;0} N = 11
Data with 11 branches
{0;0;0} N = 11
{0;0;1} N = 11
{0;0;2} N = 11
{0;0;3} N = 11
{0;0;4} N = 11
{0;0;5} N = 11
{0;0;6} N = 11
{0;0;7} N = 11
{0;0;8} N = 11
{0;0;9} N = 11
{0;0;10} N = 11
Data with 11 branches
{0;0;0} N = 11
{0;0;1} N = 11
{0;0;2} N = 11
{0;0;3} N = 11
{0;0;4} N = 11
{0;0;5} N = 11
{0;0;6} N = 11
{0;0;7} N = 11
{0;0;8} N = 11
{0;0;9} N = 11
{0;0;10} N = 11
Data with 121 branches
{0;0;0;0} N = 11
{0;0;0;1} N = 11
{0;0;0;2} N = 11
{0;0;0;3} N = 11
{0;0;0;4} N = 11
{0;0;0;5} N = 11
{0;0;0;6} N = 11
{0;0;0;7} N = 11
{0;0;0;8} N = 11
{0;0;0;9} N = 11
{0;0;0;10} N = 11
{0;0;1;0} N = 11
{0;0;1;1} N = 11
{0;0;1;2} N = 11
{0;0;1;3} N = 11
{0;0;1;4} N = 11
{0;0;1;5} N = 11
{0;0;1;6} N = 11
{0;0;1;7} N = 11
{0;0;1;8} N = 11
{0;0;1;9} N = 11
{0;0;1;10} N = 11
{0;0;2;0} N = 11

Data matching with multiple objects
Item Access solution



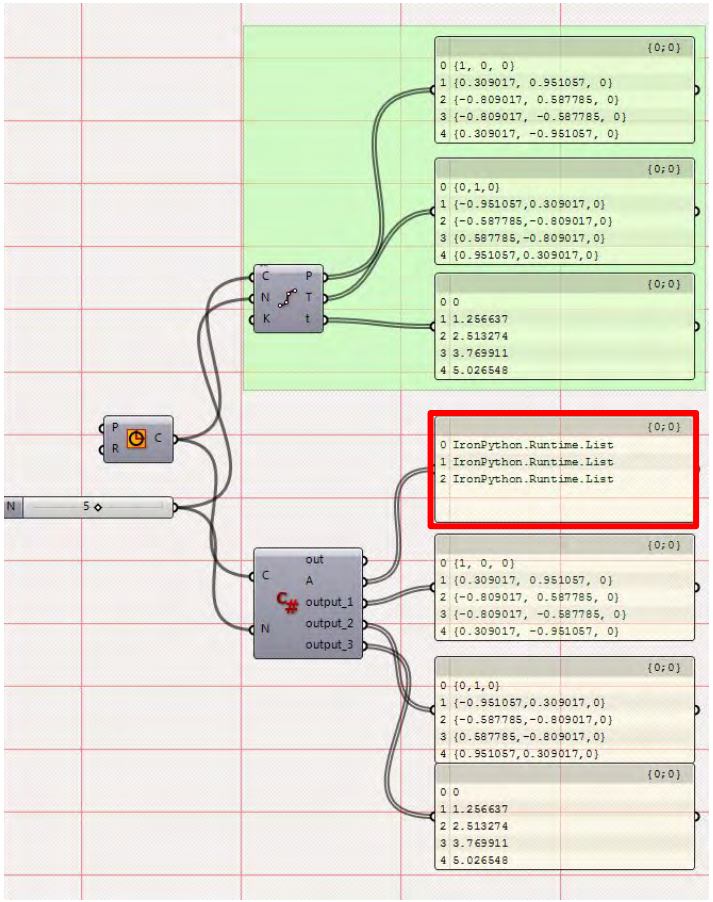
Script Editor

Script component: C# Script

```
using

/// <summary>
/// This class will be instantiated on demand by the Script component.
/// </summary>
public class Script_Instance : GH_ScriptInstance
{
  Utility functions

  Members

  /***/
  private void RunScript(Point3d pts, double radius, ref object A, ref object B)
  {
    Rhino.NodeInCode.ComponentFunctionInfo d = Rhino.NodeInCode.Components.FindComponent("Circle");

    string[] error;
    object[] drawCircle = d.Evaluate(new object[]{pts, radius}, false, out error);

    Rhino.NodeInCode.ComponentFunctionInfo e = Rhino.NodeInCode.Components.FindComponent("TrimTree");

    object[] output = e.Evaluate(new object[]{drawCircle[0]}, true, out error);

    A = output[0];

  }

  // <Custom additional code>

  // </Custom additional code>
}
```

Cache Recover from cache OK

IronPython to C#




```
public abstract Object[] Evaluate(
        IEnumerable args,
        bool keepTree,
        out string[] warnings
)
```

Parameters

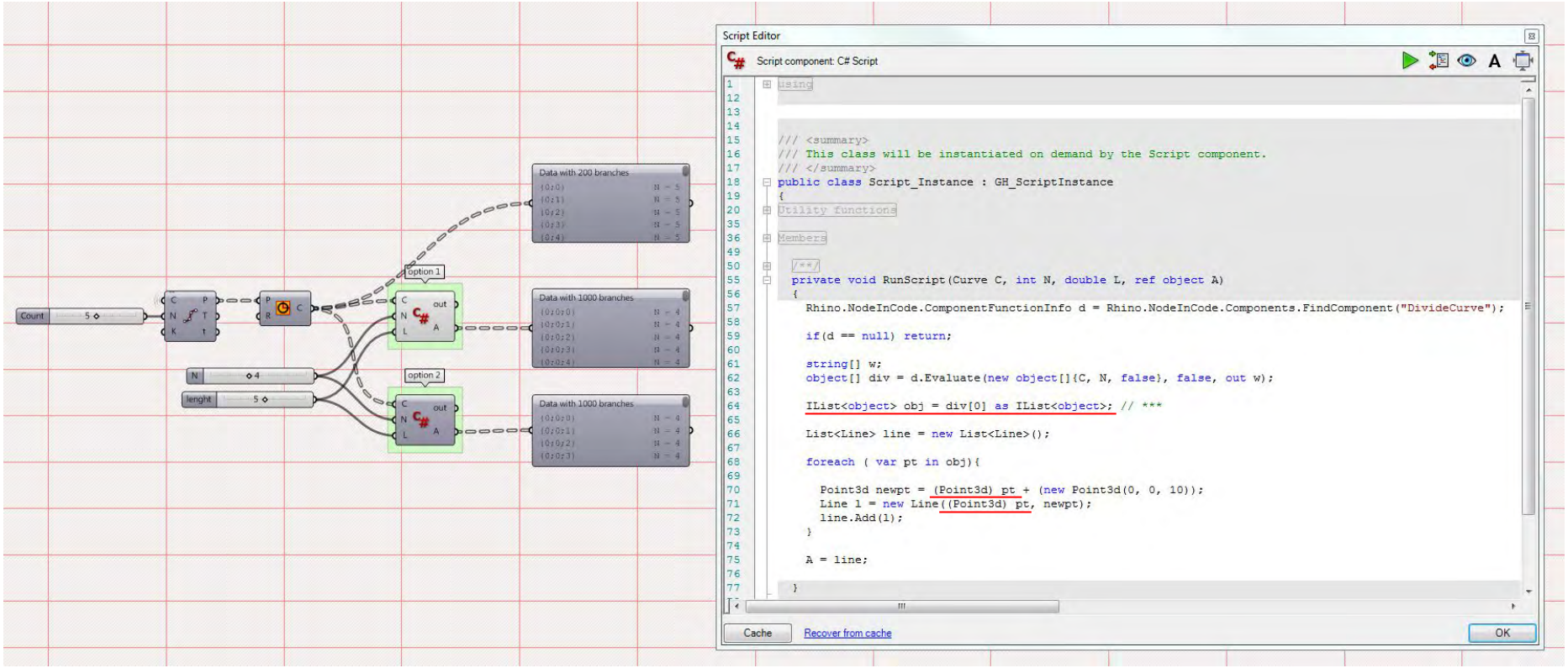
args

Type: System.Collections.IEnumerable

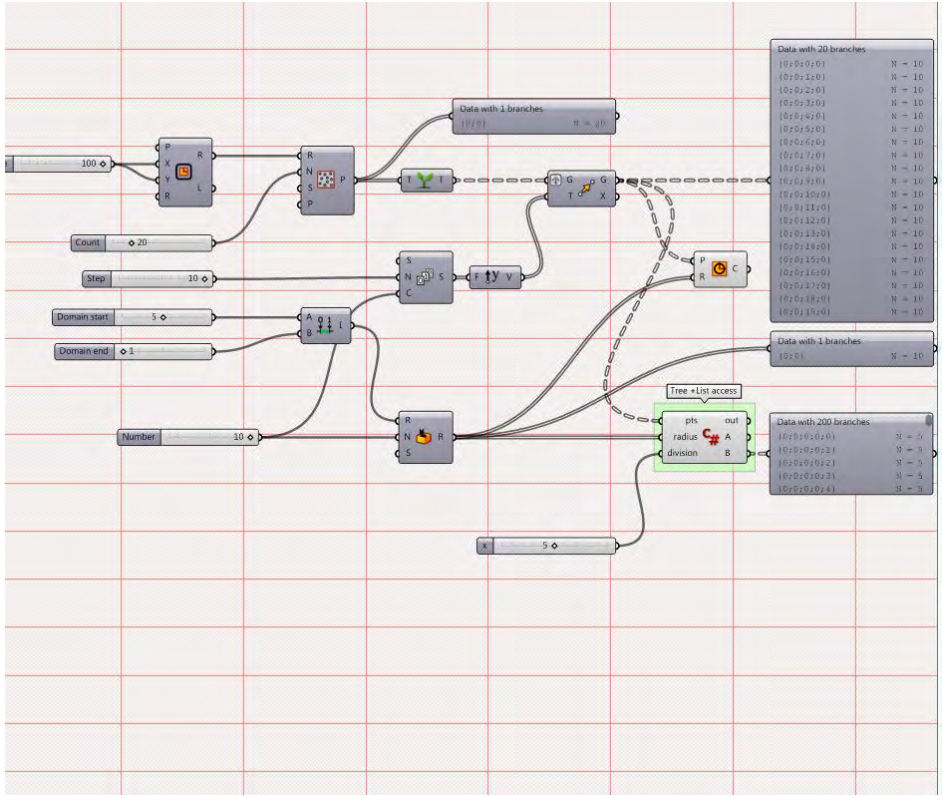
The arguments list. Each item is assigned to each input param, in order.

IronPython.Runtime.List implements the following interfaces:
IEnumerable<object> / ICollection<object> / IList<object>

NodeInCode - work with List



NodeInCode - work with DataTree

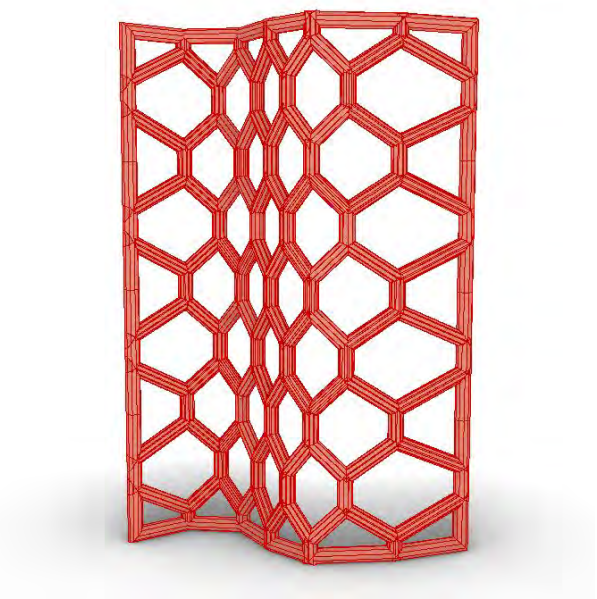


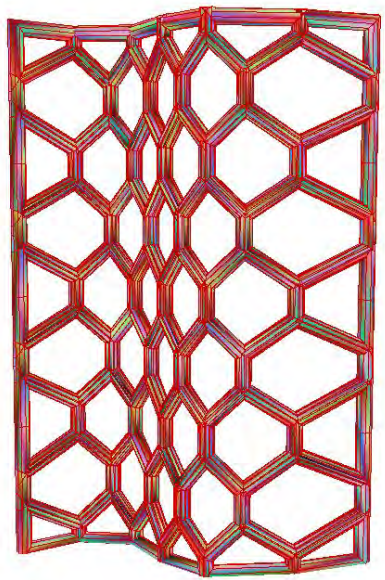
Script Editor

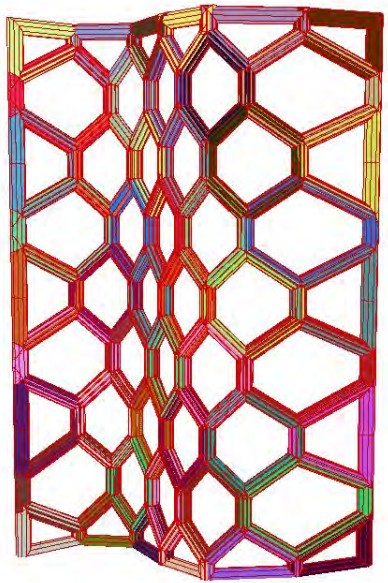
Script component: C# Script

```
using

/// <summary>
/// This class will be instantiated on demand by the Script component.
/// </summary>
public class Script_Instance : GH_ScriptInstance
{
Utility functions

Members

/**/
private void RunScript(DataTree<Point3d> pts, List<double> radius, int division, ref object A, ref object
{
  Rhino.NodeInCode.ComponentFunctionInfo d = Rhino.NodeInCode.Components.FindComponent("Circle");

  string[] error;
  object[] drawCircle = d.Evaluate(new object[]{pts, radius}, true, out error);

  DataTree<object> tree = drawCircle[0] as DataTree<object>;

  DataTree<Circle> circletree = new DataTree<Circle>();

  foreach ( var p in tree.Paths){
    foreach ( var c in tree.Branch(p)){
      Circle ci = (Circle) c;

      circletree.Add(ci, p);
    }
  }

  A = circletree;

  Rhino.NodeInCode.ComponentFunctionInfo e = Rhino.NodeInCode.Components.FindComponent("DivideCurve");
  object[] result2 = e.Evaluate(new object[]{circletree, division,}, true, out error);

  B = result2[0];



}
```

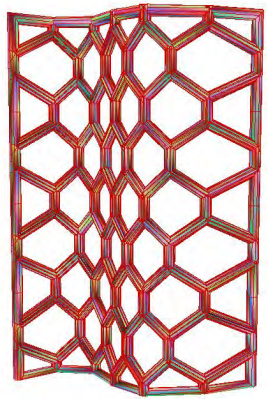
Example - Using third party plug-in(s)
- Hybrid NodeInCode workflow / scenario

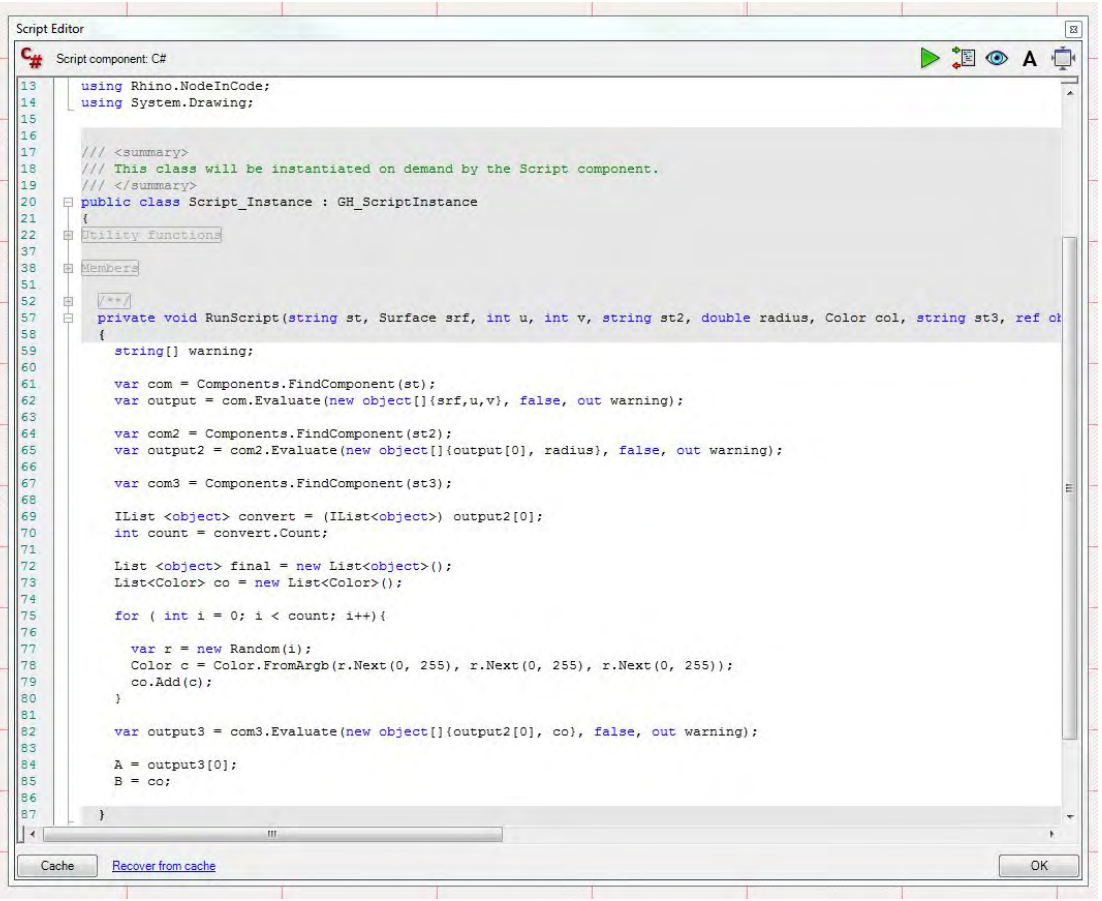






NodeInCode method - data matching issue





Shape Grammar

Shape Grammar is computation theory that defines a formalism to represent visual and spatial thinking, and was introduced by George Stiny and James Gips in 1972. Shape grammars are descriptive and generative, combining a set of shape rules and transformation methods to create geometric patterns.

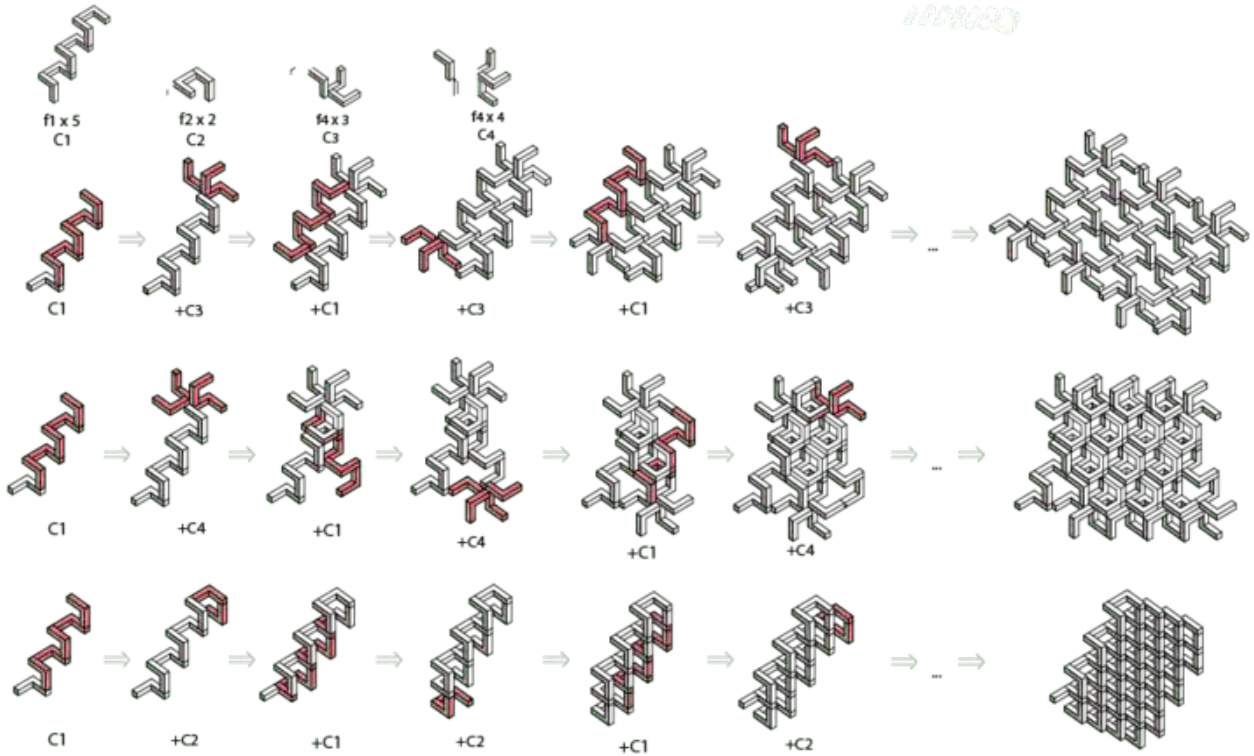


image source: Google

Shape Grammar Introduction
https://www.youtube.com/watch?v=faycjymzTzY

MIT
http://www.mit.edu/~tknight/IJDC/page_introduction.htm#:~:text=A%20shape%20grammar%20is%20a,forms%20of%20the%20generated%20designs.

L system (Lindenmayer-system)

Example 1: Algae [edit]

Lindenmayer's original L-system for modelling the growth of algae.

variables : A B
constants : none
axiom : A
rules : (A → AB), (B → A)

which produces:

n = 0 : A
n = 1 : AB
n = 2 : ABA
n = 3 : ABAAB
n = 4 : ABAABABA
n = 5 : ABAABABAABAAB
n = 6 : ABAABABAABAABABAABABA
n = 7 : ABAABABAABAABABAABABAABAABABAABAAB

Example 1: Algae, explained [edit]

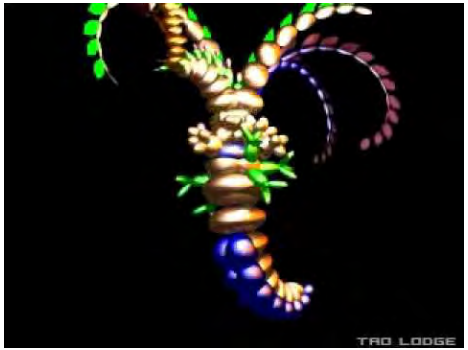
```
n=0:              A             start (axiom/initiator)
                 / \
n=1:            A   B           the initial single A spawned into AB by rule (A → AB), rule (B → A) couldn't be applied
               /|    \
n=2:          A B     A         former string AB with all rules applied, A spawned into AB again, former B turned into A
             / | |     | \
n=3:        A B A     A B       note all A's producing a copy of themselves in the first place, then a B, which turns ...
           / | | | \   | \ \
n=4:      A B A A B   A B A     ... into an A one generation later, starting to spawn/repeat/recurse then
```

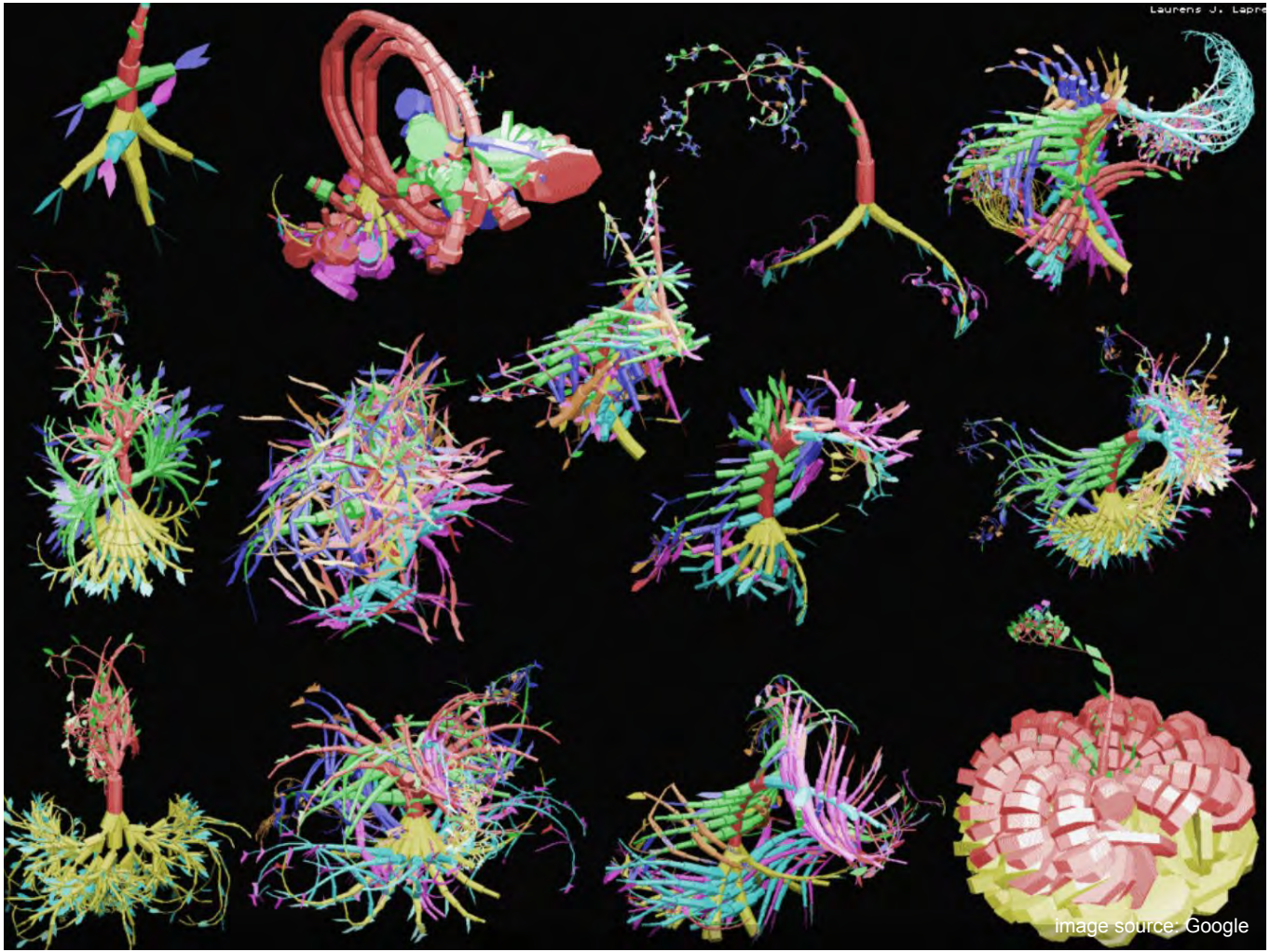
https://en.wikipedia.org/wiki/L-system

L-parser

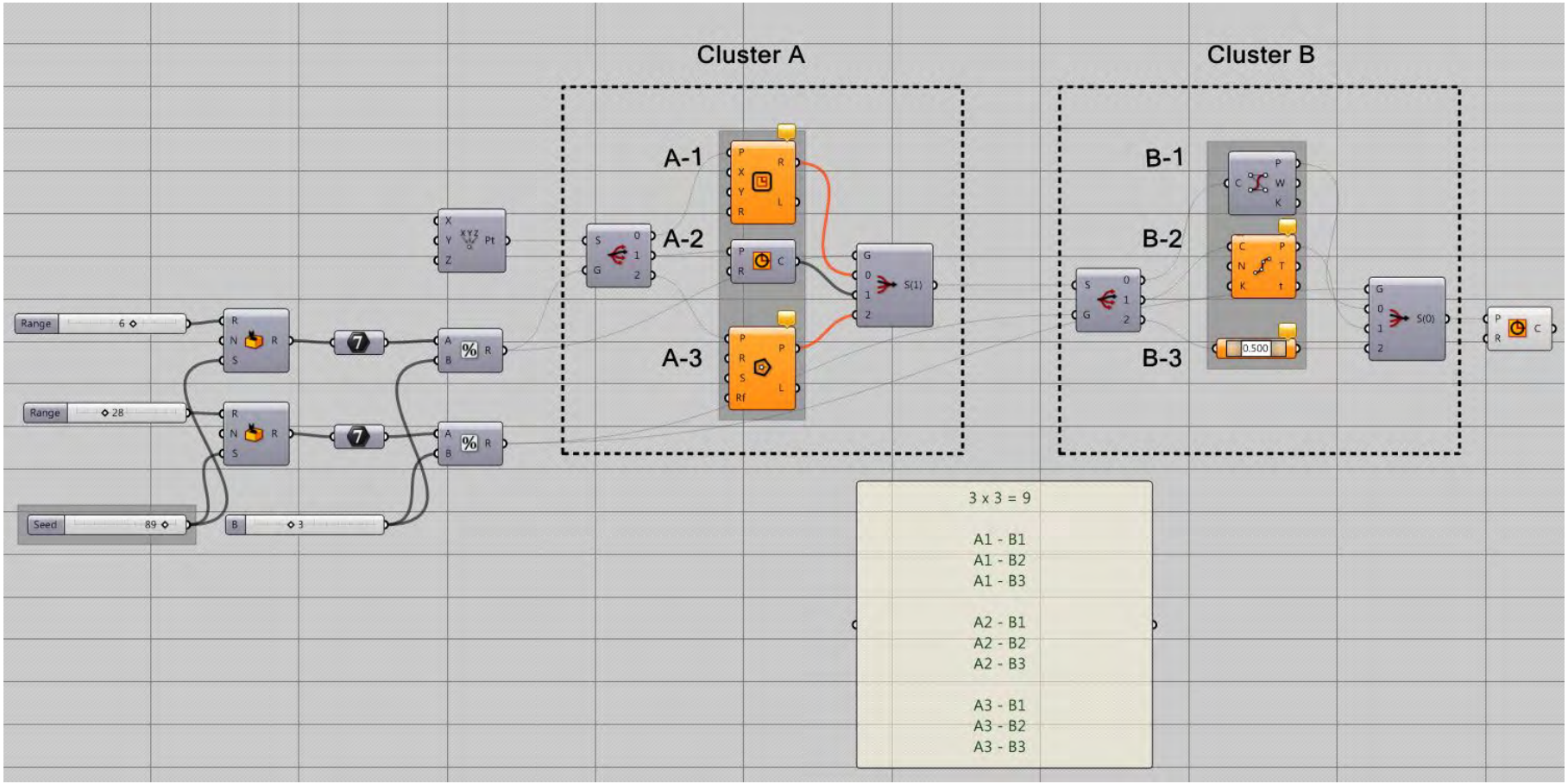
Lsystem parser allows mutations to take place during the growing of a form. These mutations can change the form slightly or dramatically. It allows a form to be created and then create a whole series of forms all clearly descendant from the original Lsystem.

http://laurenslapre.nl/ViewLparser/ViewLparser.html





Example - random composition



Shape Grammar based on iteration and combination

Circle + End Points
Rectangle + Control Points
Circle + End Points

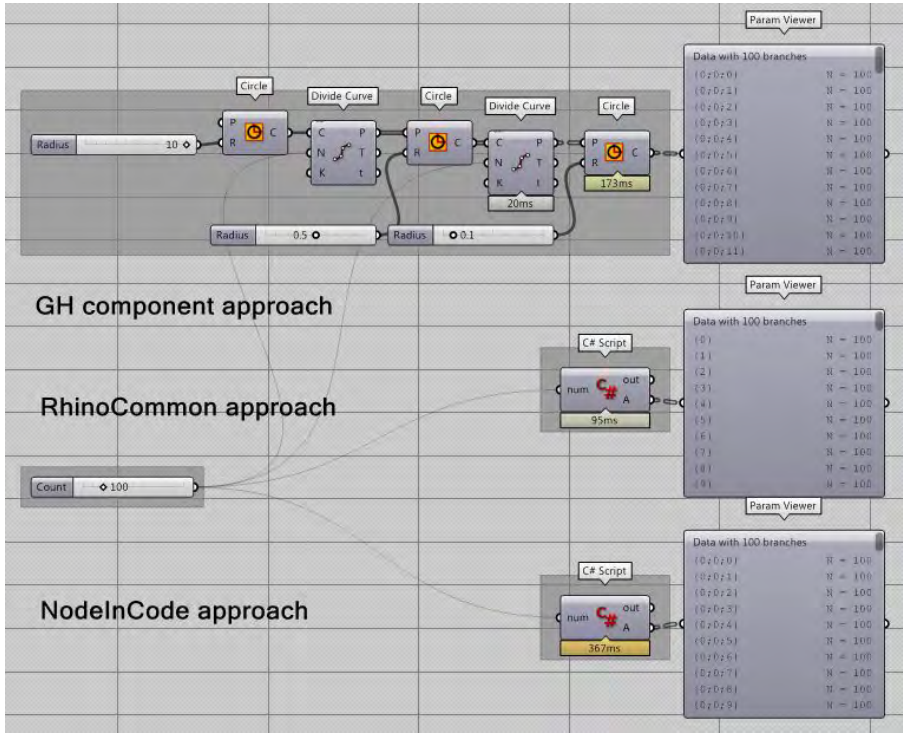
Circle + End Points
Rectangle + Control Points
Polygon + End Points

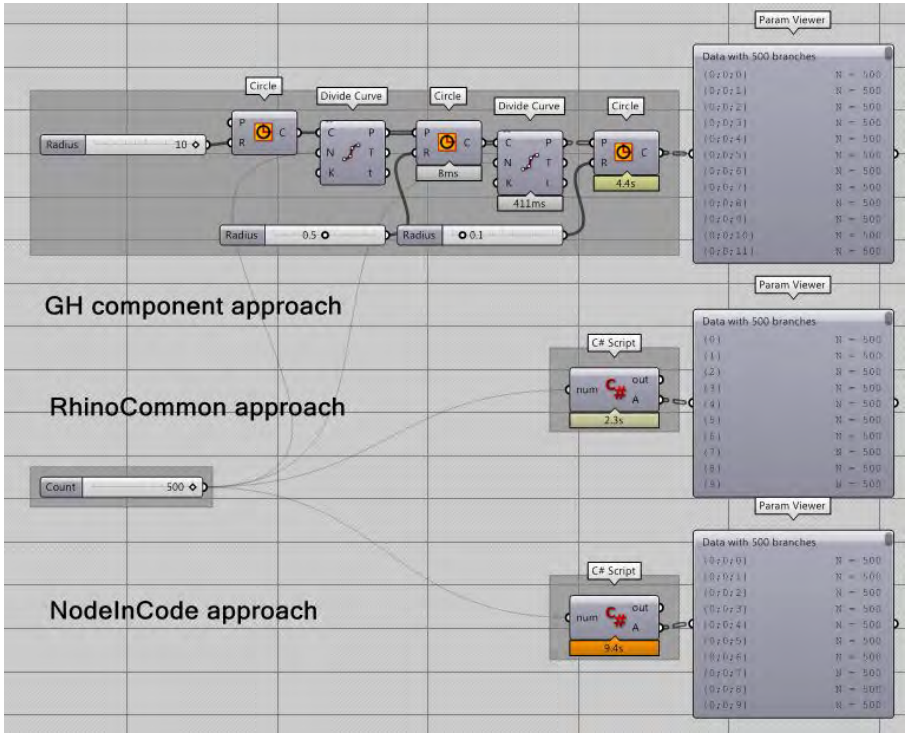
Polygon + End Points
Rectangle + Control Points
Polygon + End Points

Rectangle + Control Points
Polygon + Divide Curve
Rectangle + Control Points

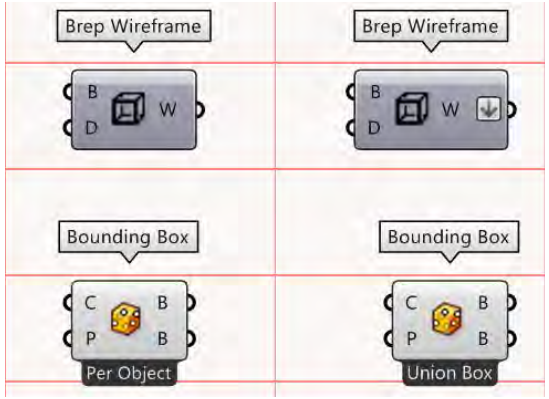
Polygon + Divide Curve
Rectangle + Control Points
Polygon + Divide Curve

NodeInCode is very convenient, but any downside?





NodeInCode - quick summary

Pros	Cons
* More intuitive and similar to component assembly principle in GH * Easy to compose a series of Rhinocommon function together * Easy access to plug-in functionality * Easy to conduct interoperability with other platforms (if the function exists) * Able to access components not documented in Rhinocommon	* Cannot access parameters in components  * Some components are not supported * Not suitable for external use * Performance

Thank you very much!

If you have any questions, please feel free to drop me an email: ping-hsiang@dezact.org