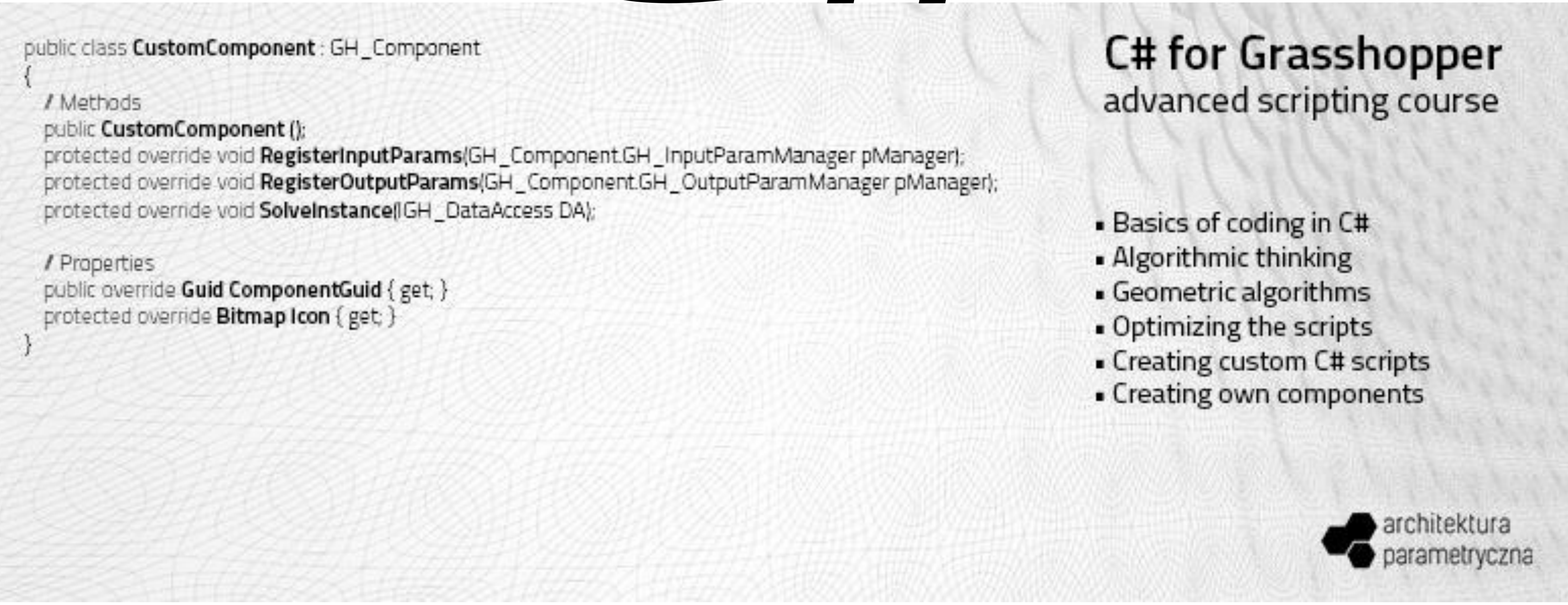
C#



```
public class CustomComponent : GH_Component
{
   / Methods
   public CustomComponent ();
   protected override void RegisterInputParams(GH_Component.GH_InputParamManager pManager);
   protected override void RegisterOutputParams(GH_Component.GH_OutputParamManager pManager);
   protected override void SolveInstance(IGH_DataAccess DA);

   / Properties
   public override Guid ComponentGuid { get; }
   protected override Bitmap Icon { get; }
}
```

C# for Grasshopper
advanced scripting course

- Basics of coding in C#
- Algorithmic thinking
- Geometric algorithms
- Optimizing the scripts
- Creating custom C# scripts
- Creating own components

architektura
parametryczna

.NET

Framework for Windows applications

CLR

Before C# language there was C/C++ languages. Compiler compile code written in Native Language for the machine to understand. In this way for instance, applications for Windows may not run on Linux.

But in C# code is not translated to Native Code. The idea is borrowed from Java, when code is translated to Byte Code, but in C# it is called IL Code (**Intermediate Language**).

It is independent on computer on which it is running.
Now we need something that translates the code the code to Native Code and that is the job of CLR (**common language runtime**). This process is called **Just-In-Time Compilation** (JIT).

Class Library

The .NET Framework class library is a library of classes, interfaces, and value types that provide access to system functionality. It is the foundation on which .NET Framework applications, components, and controls are built.

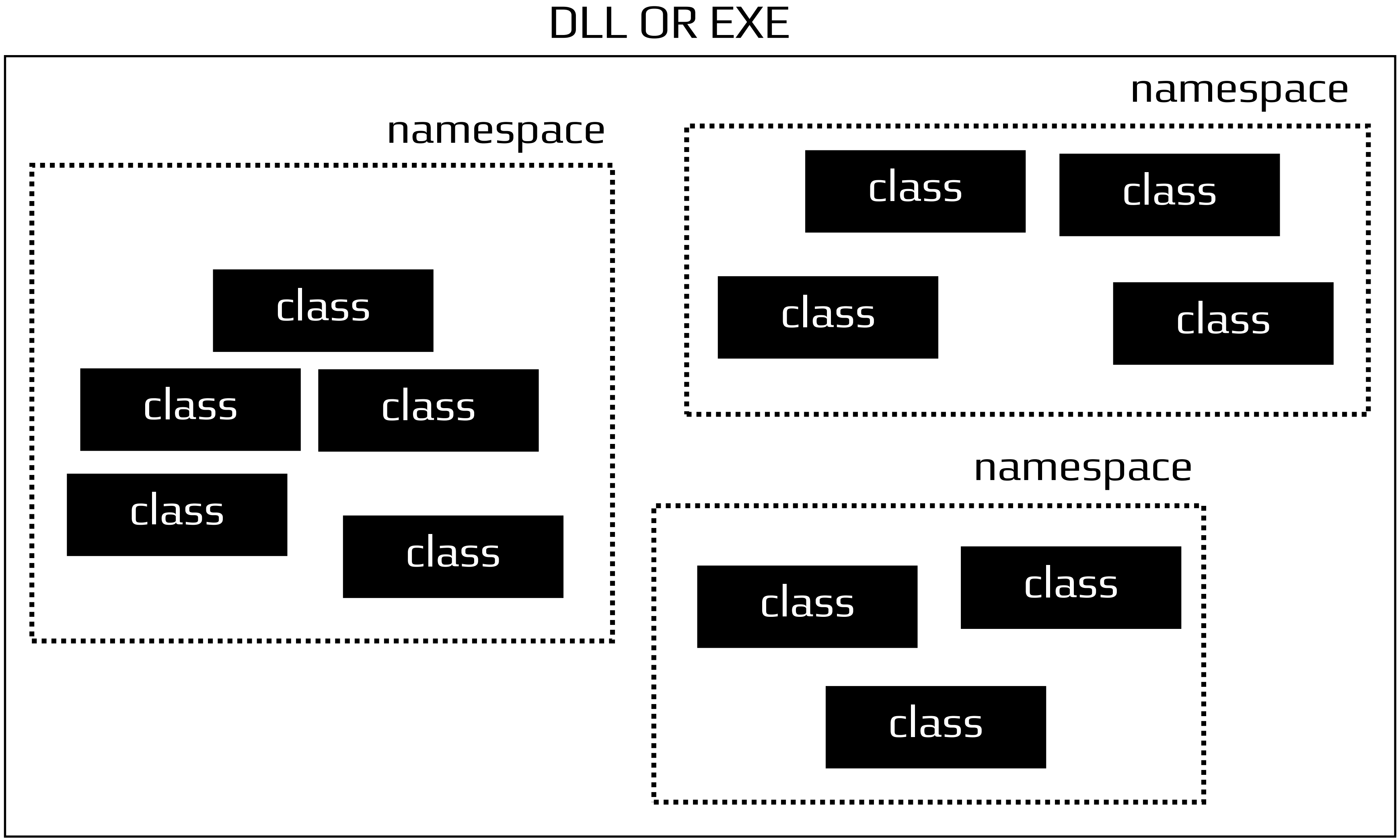
System
System.Drawing
System.Linq
System.Threading
System.Runtime
System.Windows
System.Net
Microsoft.CSharp
...

.NET LANGUAGES

C# - F# - VB.NET - J# - P# ...

Programming Languages to write Windows applications.

ASSEMBLY


DLL OR EXE

namespace

class

class

class

class

class

namespace

class

class

class

class

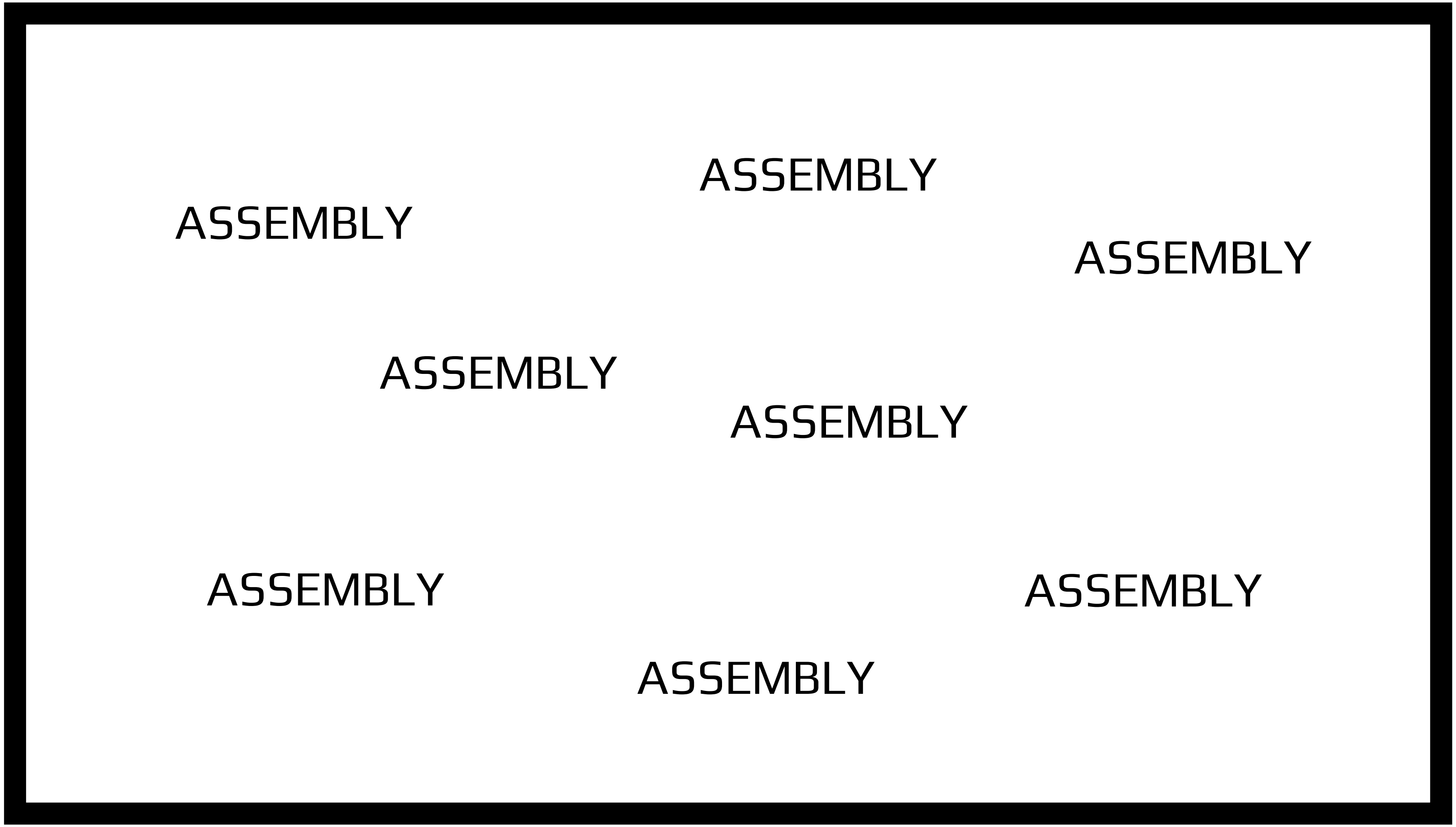
namespace

class

class

class

APPLICATION



ASSEMBLY

ASSEMBLY

ASSEMBLY

ASSEMBLY

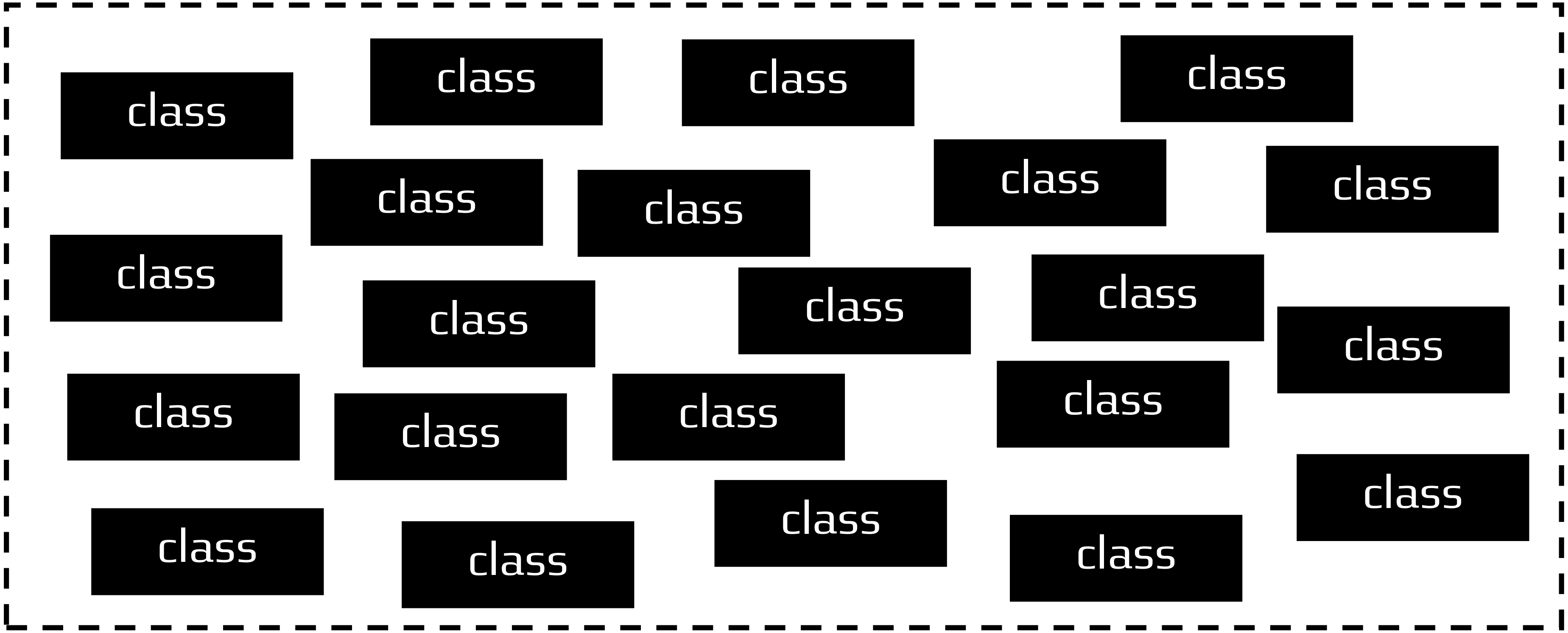
ASSEMBLY

ASSEMBLY

ASSEMBLY

ASSEMBLY

ARCHITECTURE .NET



class

class

class

class

class

class

class

class

class

class

class

class

class

class

class

class

class

class

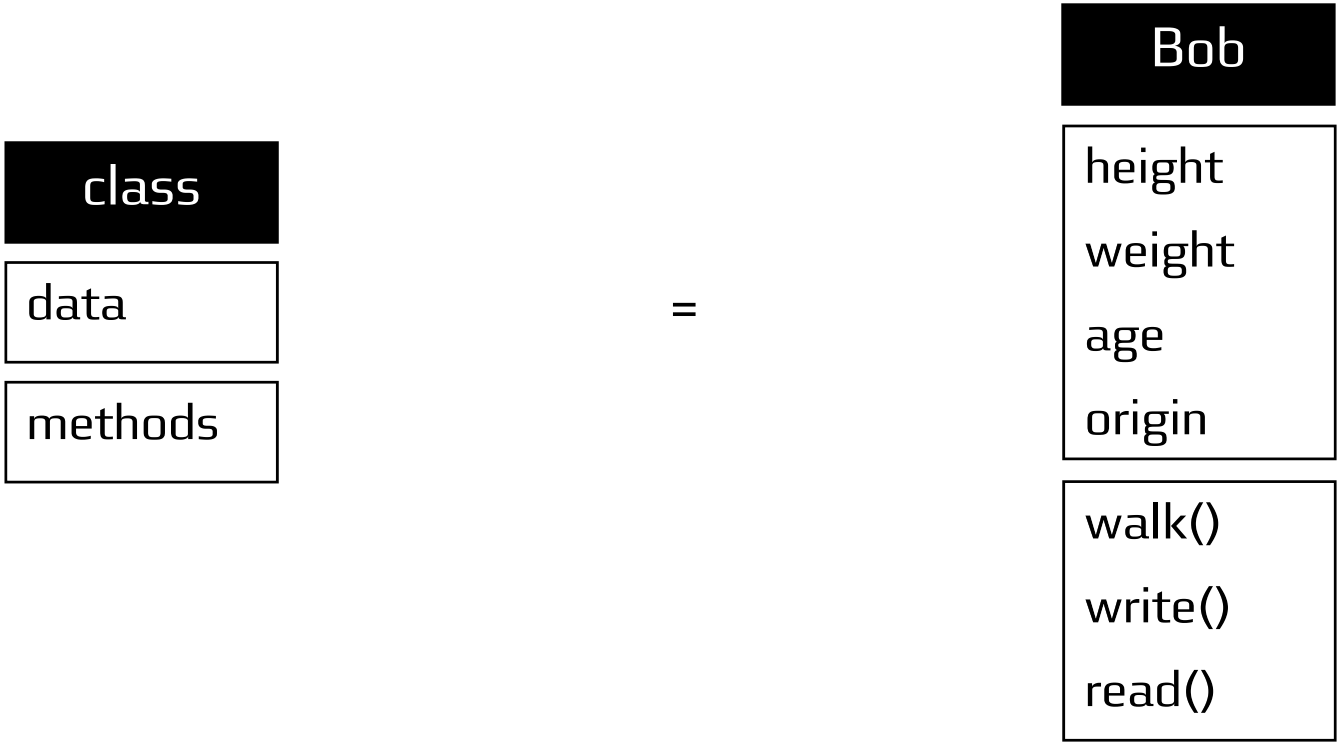
class

class

class

class

CLASS



class

data

methods

=

Bob

height

weight

age

origin

walk()

write()

read()

SYNTAX

COMMENTS

C# TYPE	SYNTAX
SINGLE LINE COMMENT Comment many lines Ctrl + K + C Uncomment many lines Ctrl + K + U	// text
MUTLI LINE COMMENT	/* text */
XML COMMENT	/// text

PRIMITIVE TYPES

	C# TYPE	.NET TYPE	BYTES	RANGE
INTEGRAL NUMBERS	byte	Byte	1	0 to 255
	short	Int16	2	-32.768 to 32.767
	int	Int32	4	-2.1B to 2.1B
	long	Int64	8	...
REAL NUMBERS	float	Single	4	-3.4 x 10^{36} to 3.4 x 10^{36}
	double	Double	8	...
	decimal	Decimal	16	-7.9 x 10^{28} to 7.9 x 10^{28}
CHARACTER	char	Char	2	Unicode Characters
BOOLEAN	bool	Boolean	1	true/false

NON-PRIMITIVE TYPES

C# TYPE	SYNTAX
VALUE TYPES USER - DEFINED	enum struct
REFERENCED TYPES - PREDEFINED	string object
REFERENCED TYPES - USER DEFINED	class Array delegate interface

OPERATORS

C# TYPE	SYMBOL
SIGN OPERATORS	+ -
ARITHMETIC	+ - * / %
LOGICAL	& | ^ ! ~ && || true false
STRING CONCATENATION	+
INCREMENT, DECREMENT	++ --
SHIFT	<< >>
RELATIONAL	== != < > <= >=
ASSIGNMENT	= += -= *= /= %= &= |= ^= <<= >>=
MEMBER ACCESS	.
INDEXING	[ ]
CAST	( )
TERNARY	?:
DELEGATE CONCATENATION AND REMOVAL	+ -
OBJECT CREATION	new
TYPE INFORMATION	as is sizeof typeof
OVERFLOW EXCEPTION CONTROL	checked unchecked
INDIRECTION AND ADDRESS	* -> [ ] &
LAMBDA	=>

CONDITIONALS

C# TYPE	SYNTAX

IF/ELSE	int hour = 10;
	
	if(hour > 0 && hour < 12)
	    ...
	else if(hour >= 12 && hour < 18)
	    ...
	else
	    ...

CONDITIONAL OPERATOR	bool isGoldCustomer = true;
	
	double price = (isGoldCustomer) ? 19.95 : 29.95;

SWITCH/CASE

```
int minute = 10;

switch (minute)
{

    case (60):
        ...
        break;

    case (30):
        ...
        break;

    case (10):
    case (20):
        ...
        break;

    default:
        ...
        break;

}
```

ITERATIONS

LOOP TYPE	SYNTAX

FOR
iterating number of times
	for (var i = 0; i < 10; i++)
	{
	...
	}

FOREACH
iterating over a list of objects, but they cannot be changed
	foreach (var numbers in numbers)
	{
	...
	}

WHILE
when you do not know how many iterations you will have
	while (i < 10)
	{
	...
	i++;
	}

DO-WHILE	do
	{
	...
	i++;
	} while (i < 10);

LOOP STATEMENTS

STATEMENT	SYNTAX
BREAK LOOP JUMPS OUT OF THE LOOP	break;
CONTINUE JUMPS TO THE NEXT ITERATION	continue;

ARRAYS

ARRAYS	SYNTAX

SINGLE DIMENSION ARRAY	var numbers = new int[5];
	var numbers = new int[5] { 1,2,3,4,5 };

Access/assign one element	var element = numbers[0];

MULTIDIMENSION ARRAYS
RECTANGULAR (MATRICES)
	var matrix = new int[3, 5];
	var matrix = new int[3, 5]
	{
	{ 1, 2, 3, 4, 5 },
	{ 6, 7, 8, 9, 10 },
	{ 11, 12, 13, 14, 15 } };

Access/assign one element	var element = matrix[0,0];

MULTIDIMENSION ARRAYS
JAGGED
	var array = new int[3][];
	array[0] = new int[4];
	array[1] = new int[5];
	array[2] = new int[3];

Access/assign one element	array[0][0] = 1;

C# ARRAY METHODS	SYNTAX
ARRAY LENGTH	numbers.Length();

GET MAXIMUM / MINIMUM ELEMENT	int a = numbers.Max();
	int b = numbers.Max();

ADD TWO ARRAYS TOGETHER	var array1 = new int[] {1,2,3};
	var array2 = new int[] {4,5,6};
	var zip = array1.Zip(array2, (a,b) => (a+b));

GET INDEX OF ELEMENT	Array.IndexOf(numbers, 3);
SET PORTION OF ELEMENTS TO ZERO, FALSE OR NULL	Array.Clear(numbers, 0, 2);
COPY PORTION OF ARRAY TO ANOTHER ARRAY	Array.Copy(numbers, another, 3);
SORT ELEMENTS IN ARRAY	Array.Sort(numbers);
REVERSE ELEMENTS IN ARRAY	Array.Reverse(numbers);

LISTS

C# TYPE	SYNTAX
LIST	
	var numbers = new List <int> ();
Access/assign one element	var numbers = new List <int> {1,2,3,4};
Copy List	var numbers = new List <int> (anotherList);

C# LIST METHODS	SYNTAX
ADD ELEMENT	numbers.Add(1);
ADD MORE THAN ONE ELEMENT	numbers.AddRange(new List <int> {4,5,6,7});
DELETE FIRST MATCHING ELEMENT OTHERWICE USE THIS IN FOR LOOP	numbers.Remove(1);
DELETE ELEMENT AT INDEX	numbers.RemoveAt(0);
GET INDEX OF AN ELEMENT OTHERWISE IT RETURNS -1	int index = numbers.IndexOf(1);
CHECK IF LIST CONTAINS AND ELEMENT	bool flag = numbers.Contains(1);
LIST LENGTH	int howmany = numbers.Count;

TIME

C# TYPE	SYNTAX
DATETIME	var dateTime = new DateTime (2015,1,1);

immutable
methods
	var now = DateTime.Now;
	var tomorrow = now.AddDays(1);
	var tomorrow = now.AddDays(-1);

convert to string	string a = now.ToLongDateString();
	string b = now.ToShortDateString();
	string c = now.ToLongTimeString();
	string d =now.ToShortTimeString();
	string e =now.ToString("yy-MM-dd HH:mm");

TIMESPAN	var timeSpan = new TimeSpan(1,2,3);

immutable
methods
	var timeSpan2 = TimeSpan.FromHours(1);
	timeSpan2.Add(TimeSpan.FromMinutes(2)));
	timeSpan2.Subtract(TimeSpan.FromMinutes(2)));

properties
convert to string
parsing
	int minutes = timeSpan.Minutes;
	string a = timeSpan.ToString();
	TimeSpan b = TimeSpan.Parse("01:02:03");

STRINGS

C# TYPE	SYNTAX
FORMATING	
All lower//upper case letters	ToLower() / ToUpper()
Removes white spaces	Trim()
SEARCHING	
letter	IndexOf(‘a’)
last index	LastIndexOf(“Hello”)
SUBSTRINGS	
gets string from letter index	Substring(startIndex)
	Substring(startIndex, length)
REPLACING	
replace letter	Replace(‘a’, ‘!’)
replace string	Replace(“Point3d”, “Point3f”)
NULL CHECKING	
is empty or whitespace	String.IsNullOrEmpty(str) / String.IsNullOrWhiteSpace(str)
SPLITTING	
split by whitespaces	str.Split(‘ ‘)
CONVERT	
parse string to int	int i = int.Parse(s);
convert string to int	int j = Convert.ToInt32(s);
convert numbers to strings	int i = 1234;
	string s = i.ToString();

FILE

C# TYPE	SYNTAX

FILE
 	
static methods
 	
used if you want to execute small number of operations
	File.Copy(@"C:\Users\pves\Desktop\Folder1\Hello.txt", @"C:\Users\pves\Desktop\Folder2\Hello.txt",true);
	
	File.Delete(@"C:\Users\pves\Desktop\Folder2\Hello.txt");
	
	var path = @"C:\Users\pves\Desktop\Folder1\Hello.txt";
	
	File.Exists(path)

FILE INFO
 	
instance methods
where you have to create new object each command
	var fileInfo = new FileInfo(path);
	
	fileInfo.CopyTo("...");
	
	fileInfo.Delete();
	
	if (fileInfo.Exists)

DIRECTORY

C# TYPE	SYNTAX

DIRECTORY
static methods
used if you want to execute small number of operations
	Directory.CreateDirectory(@"C:\Users\pves\Desktop\Folder1\temp");
	var files = Directory.GetFiles(@"C:\Users\pves\Desktop\Folder1", "*.txt", SearchOption.AllDirectories);
	var directories = Directory.GetDirectories(@"C:\Users\pves\Desktop", "*.*", SearchOption.AllDirectories);
	Directory.Exists("...");

DIRECTORY INFO
instance methods
where you have to create new object each command
	var directoryInfo = new DirectoryInfo("...");
	directoryInfo.GetFiles();
	directoryInfo.GetDirectories();

FUNCTIONS

C# TYPE	SYNTAX
CREATE FUNCTION	

visibility return-type name (parameters)
{
function code
}
	public void DoStuff()
	{
	Console.WriteLine("Hello");
	}
	
	public int AddNumbers(int a, int b)
	{
	int result = a + b;
	return result;
	}

CALLING FUNCTIONS	
empty function	DoStuff();
function with return type	int sum = AddNumbers(1,2);

REF VS OUT

KEYWORD	SYNTAX

REF
 	
Here a will be equal to 22 without ref keyword a would remain 20.
	int a = 20;
	SomeFunction(ref a);
	
	public void AddNumbers(ref int d) {
	    d = d + 2;
	}
	
	Console.WriteLine(a);

OUT
 	
But in out you need initialize variables and your passed variable is not taken inside the function as ref, just modified.
	int a = 20;
	SomeFunction(out a);
	
	public void AddNumbers(out int d) {
	    d = 0;
	    d = d + 2;
	}
	
	Console.WriteLine(a);

1. OUT AND REF HELPS TO PASS BY REFERENCE
2. REF IS TWO WAY FROM CALLER TO CALLEE AND BACK
3. OUT IS ONE WAY IT SENDS DATA BACK FROM CALLEE TO CALLER.

DEBUGGING

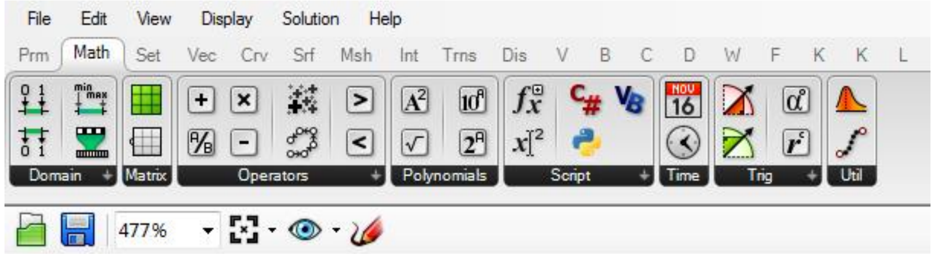
METHOD	SHORTCUT
1. YOU NEED TO BUT BREAK POINT	F9
2. RUN APPLICATION IN DEBUG MODE	F5
3. RUN APPLICATION WITHOUT DEBUG MODE	Ctrl + F5
4. CONTINUE EXECUTION (STEP OVER)	F10
5. STEP INTO THE METHOD	F11
6. STEP OUT	Shift + F11
7. TO STOP DEBUG MODE	Shift + F5

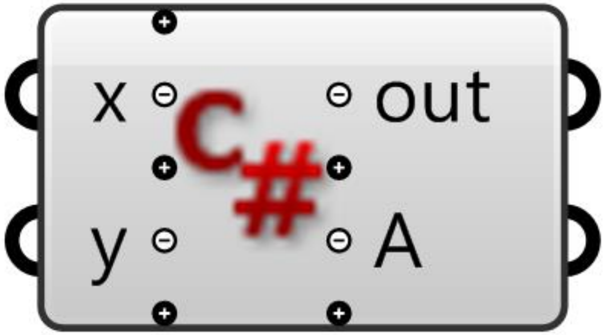
RHINOCOMMON

C#

GRASSHOPPER

C# COMPONENT





Input
Parameter "x"

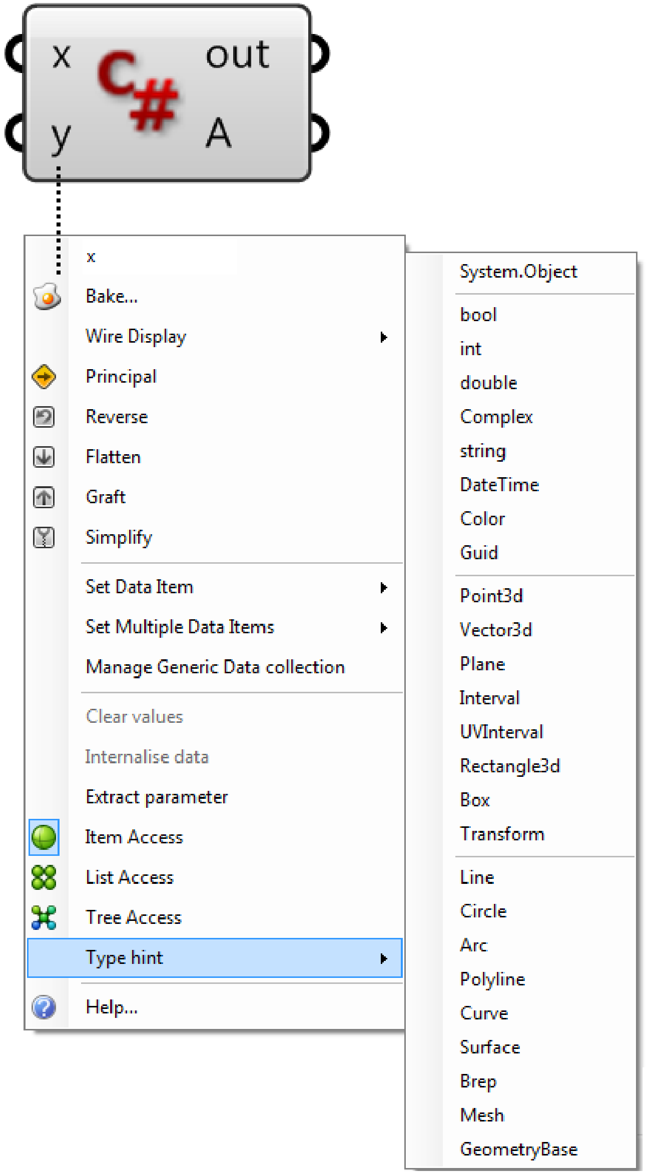
Input
Parameter "y"

String for debugging
and compiling messages

Output parameter "A"
script return values

The user can change names, types and number of inputs and outputs.

C# COMPONENT INPUTS



The ultimate base class of all classes in the .NET

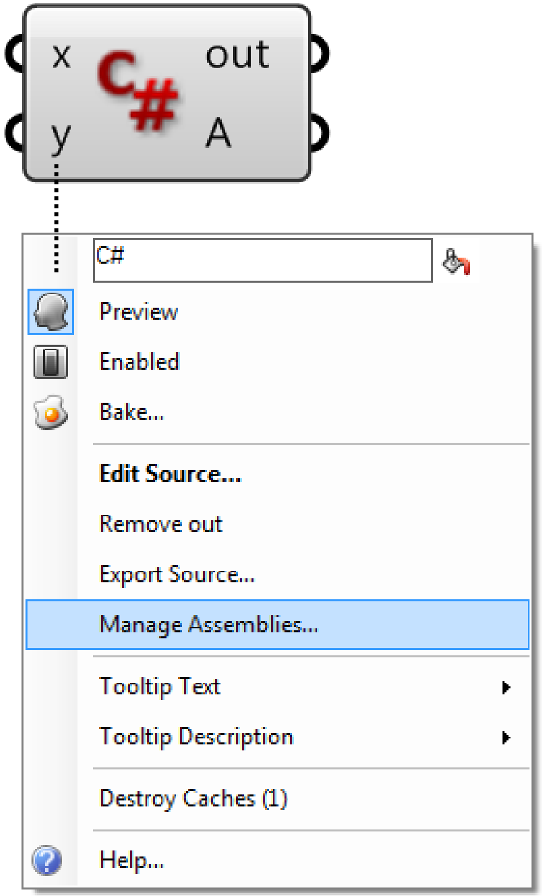
Base types

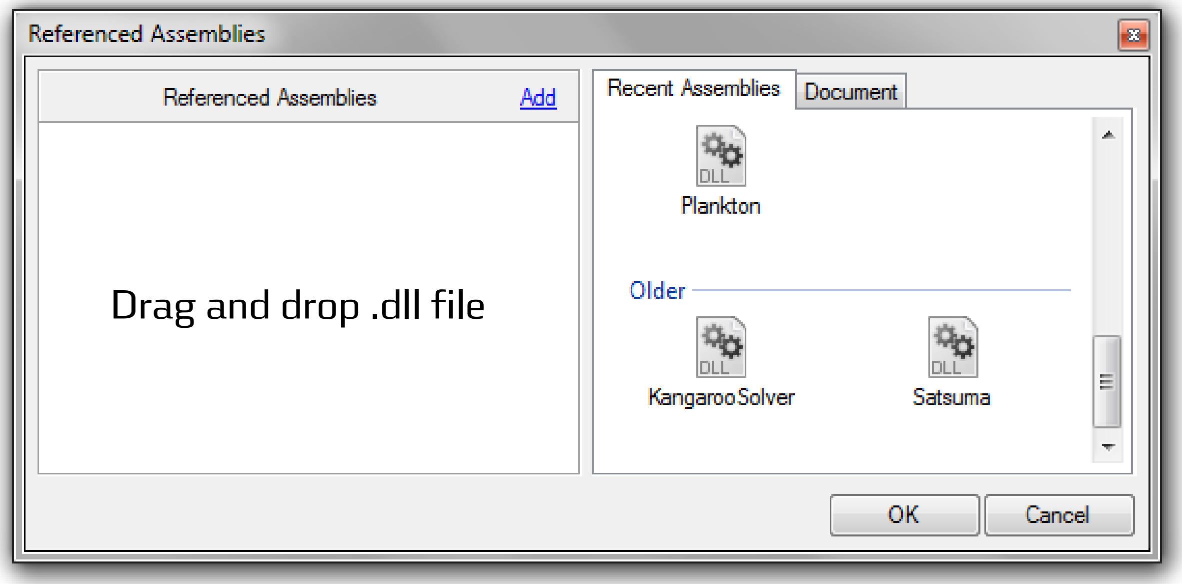
Access type

RhinoCommon types

Note: there are no types associated with output.
They are defined as generic system type “object”

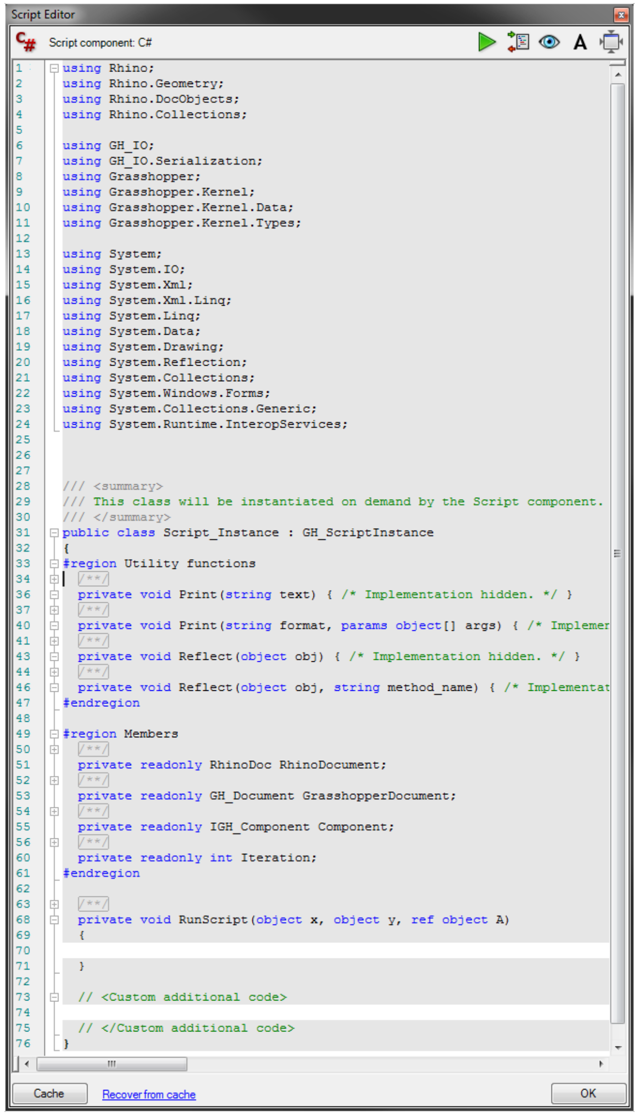
C# EXTERNAL ASSEMBLIES





Drag and drop .dll file

C# SCRIPT EDITOR



```
using Rhino;
using Rhino.Geometry;
using Rhino.DocObjects;
using Rhino.Collections;

using GH_IO;
using GH_IO.Serialization;
using Grasshopper;
using Grasshopper.Kernel;
using Grasshopper.Kernel.Data;
using Grasshopper.Kernel.Types;

using System;
using System.IO;
using System.Xml;
using System.Xml.Linq;
using System.Linq;
using System.Data;
using System.Drawing;
using System.Reflection;
using System.Collections;
using System.Windows.Forms;
using System.Collections.Generic;
using System.Runtime.InteropServices;



/// <summary>
/// This class will be instantiated on demand by the Script component.
/// </summary>
public class Script_Instance : GH_ScriptInstance
{
#region Utility functions
  /**/
  private void Print(string text) { /* Implementation hidden. */ }
  /**/
  private void Print(string format, params object[] args) { /* Implemer
  /**/
  private void Reflect(object obj) { /* Implementation hidden. */ }
  /**/
  private void Reflect(object obj, string method_name) { /* Implementat
#endregion

#region Members
  /**/
  private readonly RhinoDoc RhinoDocument;
  /**/
  private readonly GH_Document GrasshopperDocument;
  /**/
  private readonly IGH_Component Component;
  /**/
  private readonly int Iteration;
#endregion

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

  }

  // <Custom additional code>

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

Double click in the middle of the component to open C# script editor.

Imports

External .dll's that you might use in your code. Most of them are DotNET system imports, but there is also RhinoCommon and Grasshopper assemblies.

Utility functions

Print text and component error / warnings to 'out' parameters.

Members

Rhino, Grasshopper documents, component properties and iteration count.

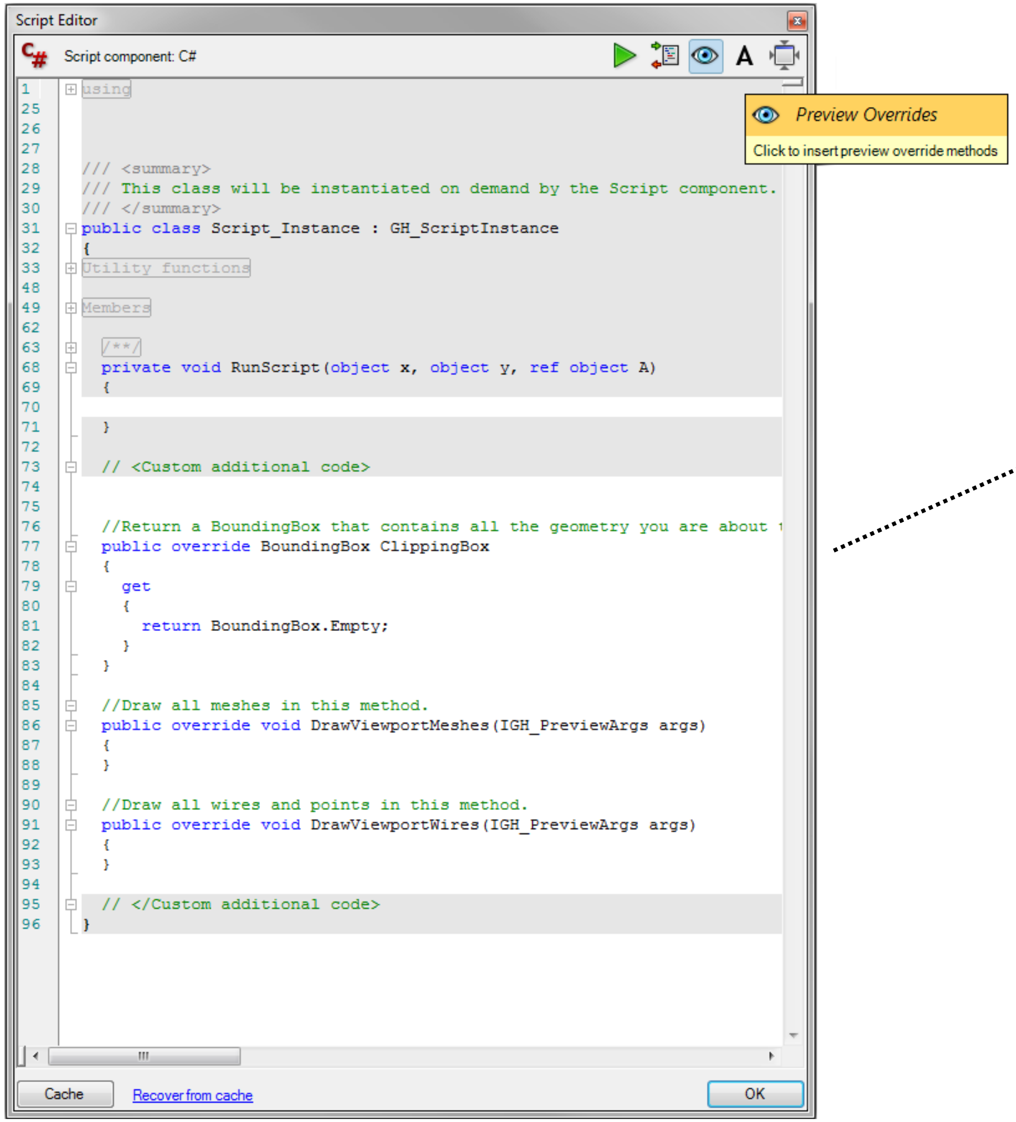
RunScript

This is the main function that the user writes the code within.

Custom additional code

Here you can write classes, function and declare variables.
Variables declared here are saved in memory.

C# SCRIPT EDITOR OVERRIDES



```
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(object x, object y, ref object A)
  {

  }

  // <Custom additional code>


  //Return a BoundingBox that contains all the geometry you are about t
  public override BoundingBox ClippingBox
  {
    get
    {
      return BoundingBox.Empty;
    }
  }

  //Draw all meshes in this method.
  public override void DrawViewportMeshes(IGH_PreviewArgs args)
  {
  }

  //Draw all wires and points in this method.
  public override void DrawViewportWires(IGH_PreviewArgs args)
  {
  }

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

Click on Preview Overrides button

override BoundingBox

You need to get 1 bounding box of all objects you display for non-flickering display.

override DrawViewportMeshes

Display objects such as Lines and Curves

override DrawViewportWires

Display objects such as Meshes, Surfaces, Breps

Note: if you are displaying geometry and outputing geometry you have to use base.DrawViewportMeshes(args); after one of the override method.

PRINT

DESCRIPTION	C# SYNTAX

Print
Command-line
	Rhino.RhinoApp.WriteLine(x.ToString());
	Rhino.RhinoApp.Write (x.ToString());

Print
Component Out
Parameter
	Print(nakedPtsB.GetType().ToString());
	Print("Number of vertices");

ITEM / LIST / TREE

DESCRIPTION	C# SYNTAX

Type hint:
Item Access
	private void RunScript(double x, double y, ref object A)
	{
	A = x + y;
	}

Type hint:
List Access
(loop through the list of items)
	private void RunScript(List<double> x, double y, ref object A)
	{
	for(int i = 0 ; i < x.Count; i++)
	x[i] += y;
	
	A = x;
	}

Type hint:
Tree Access
(loop through all branches, loop through specific branch items)
	private void RunScript(DataTree<double> x, double y, ref object A)
	{
	for(int i = 0 ; i < x.BranchCount; i++)
	for(int j = 0; j < x.Branch(i).Count; j++)
	x.Branch(i)[j] += y;
	
	A = x;
	}

TIMER

DESCRIPTION **C# SYNTAX**

TIMER

ExpireSolution method can be replaced with timer component.

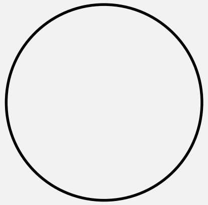
```
private void RunScript(bool reset, bool run, ref object A)
{

  if (reset)
    n = 0;
  else if(run){
    n++;
    Component.ExpireSolution(true);
  }

  A = n;

}

// <Custom additional code>
int n = 0;
```

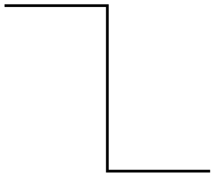
PRIMITIVE GEOMETRY

DESCRIPTION	C# SYNTAX

Point
●
	Point3d pt = new Point3d(x, y, z);
	A = pt;

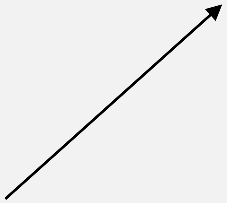
Line
 	
	Line ln = new Line(new Point3d(0, 0, 0), new Point3d(0, 0, 100));
	A = ln;

Circle

	List<Circle> circles = new List<Circle>();
	foreach (Point3d pt in pts
	circles.Add(new Circle(pt, 10))
	A = circles;

Polyline

	Polyline pl = new Polyline();
	for (int i = 0; i < 10; i ++)
	pl.Add(i, 0, Math.Pow(i, 2));
	A = pl;

VECTOR MATH

DESCRIPTION	C# SYNTAX

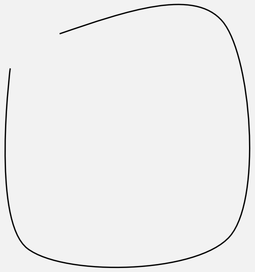
Vector3d

	//Get Unit Vector
	v1.Unitize();
	
	//Vector subtraction
	Vector3d vSubtraction = v0 - v1;
	
	//Vector dot product
	(if result in positive number then vector are in the same direction)
	double dotProduct = v0 * v1;
	
	//Scale vector
	Vector3d vScaled = 0.1 * v0;
	
	//Move a point by a vector
	Point3d movedPoint = Point3d.Origin + v0;
	
	//Distance between two points
	double distance = Point3d.Origin.DistanceTo(movedPoint);
	
	//Get Vector length
	double vectorLength = v0.Length;

NURBS CURVE

DESCRIPTION **C# SYNTAX**

NurbsCurve

```
//Create nurbs curve
int degree = 3;
NurbsCurve nc = NurbsCurve.Create(true, degree, CPoints);

//Change weight
int index = 3;
nc.Points.SetPoint(index, CPoints[index].X, CPoints[index].Y, CPoints[index].Z,
weight);

//Divide curve
Point3d[] points;
nc.DivideByCount(divisions, true, out points);

//Point on curve
Point3d pt = new Point3d();
pt = crv.PointAtNormalizedLength(t);

 //Change domain
 crv.Domain = new Interval(0, 1);
```

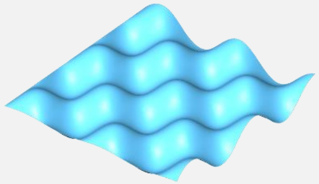


SURFACE

DESCRIPTION **C# SYNTAX**

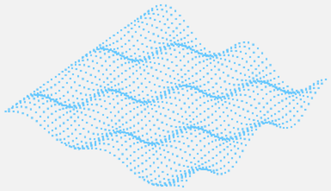
Surface

create surface from points



```
//Create grid of points
List<Point3d> Pts = new List<Point3d>();
for (int i = 0; i < n1; i++)
   for (int j = 0; j < n2; j++)
      Pts.Add(new Point3d(i, j, Math.Cos(i) * Math.Sin(j)));
//Create surface
NurbsSurface surface = Rhino.Geometry.NurbsSurface.CreateThroughPoints(Pts, n1, n2,
uDegree, vDegree, false, false);
//Change control point weight
surface.Points.SetControlPoint(6, 6, new ControlPoint(surface.Points.GetControlPoint(6,
6).Location, weight));
```

divide a surface



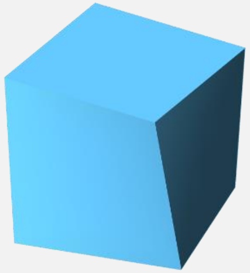
```
//Find Step - n must be greater than 1
double step = 1 / ((double) n - 1);
//Take one brep face and convert it to nurbs surface
NurbsSurface ns = brep.Faces[0].ToNurbsSurface();
//Normalize domain
ns.SetDomain(0, new Interval(0, 1));
ns.SetDomain(1, new Interval(0, 1));
//Divide surface into points
var points = new List<Point3d>();
for(double i = 0 ; i < 1; i += step)
  for(double j = 0 ; j < 1; j += step)
    points.Add(ns.PointAt(i, j));
```

BREP

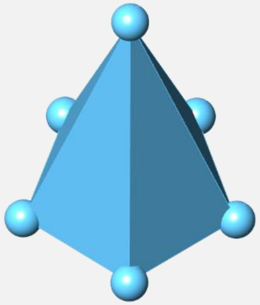
DESCRIPTION **C# SYNTAX**

Brep

create breps



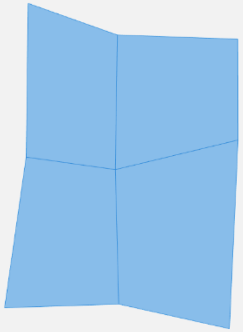
get brep properties



```
//Create cone
new Cone(new Plane(Point3d.Origin, new Vector3d(0, 1, 1)), height, radius);

//Create sphere
Sphere sphere = new Sphere(Point3d.Origin, radius);
//Get bounding box
BoundingBox bbox = sphere.BoundingBox;
//Find Box center
Point3d center = bbox.Center;

//Create from points
Brep brep = Rhino.Geometry.Brep.CreateFromBox(Corners);
//Create loft surface
Brep.CreateFromLoft(crv, Point3d.Unset, Point3d.Unset, LoftType.Normal, false);

Print("N of brep.Curves2D = " + brep.Curves2D.Count.ToString());
Print("N of brep.Curves3D = " + brep.Curves3D.Count.ToString());
Print("N of brep.Edges = " + brep.Edges.Count.ToString());
Print("N of brep.Faces = " + brep.Faces.Count.ToString());
Print("N of brep.Loops = " + brep.Loops.Count.ToString());
Print("N of brep.Surfaces = " + brep.Surfaces.Count.ToString());
Print("N of brep.Trims = " + brep.Trims.Count.ToString());
Print("N of brep.Vertices = " + brep.Vertices.Count.ToString());
```

MESH

DESCRIPTION	C# SYNTAX

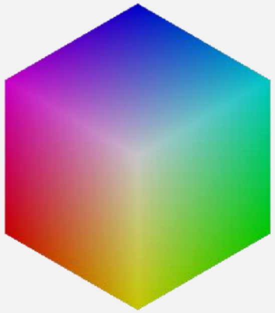
Mesh

	//Initialize mesh
	Mesh mesh = new Mesh();
	
	//Add Vertices
	mesh.Vertices.AddVertices(points);
	
	//Create faces
	mesh.Faces.AddFace(new MeshFace(0, 1, 4, 5));
	mesh.Faces.AddFace(new MeshFace(1, 2, 3, 4));
	mesh.Faces.AddFace(new MeshFace(5, 4, 7, 6));
	mesh.Faces.AddFace(new MeshFace(7, 4, 3, 8));
	
	//Clean mesh
	mesh.Vertices.CombineIdentical(true, true);
	mesh.Vertices.CullUnused();
	mesh.Weld(3.14159265358979);
	mesh.UnifyNormals();
	mesh.FaceNormals.ComputeFaceNormals();
	mesh.Normals.ComputeNormals();

POINTCLOUD

DESCRIPTION **C# SYNTAX**

PointCloud

```
PointCloud pointCloud = new PointCloud();

for (int i = 0; i < n1 ; i++)
  for(int j = 0; j < n2; j++)
    for(int k = 0; k < n3; k++)
      pointCloud.Add(new Point3d(i * dist, j * dist, k * dist),
                             Color.FromArgb(i * 2 % 255, j * 2 % 255, k * 2 % 255));

  _pointCloud = pointCloud;

// </Custom additional code> Display PointCloud
PointCloud _pointCloud = new PointCloud();

public override BoundingBox ClippingBox
{
  get { return _pointCloud.GetBoundingBox(false);}
}

public override void DrawViewportMeshes(IGH_PreviewArgs args){
  args.Display.DrawPointCloud(_pointCloud, 5);
}
```



MASS PROPERTIES

DESCRIPTION	C# SYNTAX

AREA	AreaMassProperties amp = AreaMassProperties.Compute(geo);
	Print("Area: " + amp.Area.ToString());
	
	//Find center
	A = amp.Centroid;

VOLUME	VolumeMassProperties vmp = VolumeMassProperties.Compute(geo);
	Print("Volume: " + vmp.Volume);
	
	//Find center
	B = vmp.Centroid;

TRANSFORM

DESCRIPTION **C# SYNTAX**

Point3d

```
pt.Transform(Transform.Translation(Vector3d.ZAxis)); //Transform matrix
pt += Vector3d.ZAxis; //Add vector
pt += new Point3d(0, 0, 1); //Add points
```

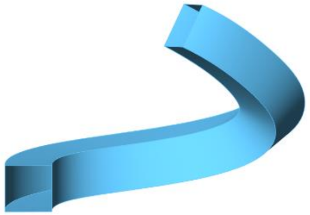
Line

```
Line newLine = line; //Line is a struct so copy lines is simple like this
newLine.Transform(Transform.Scale(newLine.PointAt(0.5), factor));
```

Ellipse

```
Ellipse c = new Ellipse(Plane.WorldXY, r1,r2);
var nurbs = c.ToNurbsCurve();
nurbs.Rotate(R * i, Vector3d.YAxis, new Point3d(X, 0, 0));
```

Curve



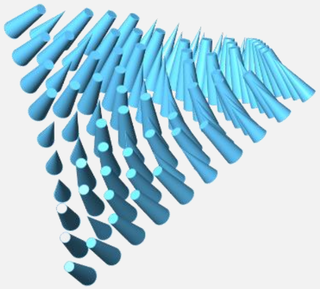
```
Plane basePlane = new Plane(profile.GetBoundingBox(false).Center,  Vector3d.ZAxis);
path.Domain = new Interval(0, 1);
List<Curve> orientedCrv = new List<Curve>();

for(int i = 0 ; i <= n; i++){
    //Get perpendicular frame and orient curves
    Plane plane;
    path.PerpendicularFrameAt((double) i / n, out plane);
    perFrames.Add(plane);
    Curve newCurve = profile.DuplicateCurve();
    newCurve.Transform(Transform.PlaneToPlane(basePlane, plane));
    orientedCrv.Add(newCurve);
}
```

TRANSFORM

DESCRIPTION **C# SYNTAX**

Brep

The same transformation method could be applied to meshes and surfaces



```
srf.SetDomain(0, new Interval(0, 1));
srf.SetDomain(1, new Interval(0, 1));
List<Plane> planes = new List<Plane>();
List<Plane> orientplanes = new List<Plane>();
List<Brep> objects = new List<Brep>();

//Get orientation plane
 Point3d pointCenter = b.GetBoundingBox(true).Center;
  Point3d pointLowest = b.GetBoundingBox(true).Min;
  Point3d pointLowCen = new Point3d(pointCenter.X, pointCenter.Y, pointLowest.Z);
  orientplanes.Add(new Plane(pointLowCen, Vector3d.ZAxis));

for(int i = 0; i < n; i++){
  for(int j = 0; j < n; j++){
    //Get Planes
    Plane plane;
    srf.FrameAt((double) i / n, (double) j / n, out plane);
    planes.Add(plane);
    //Orient breps
    Brep newBrep = breps[0].DuplicateBrep();
    newBrep.Transform(Transform.PlaneToPlane(orientplanes[0], plane));
    objects.Add(newBrep);
 }
}
```

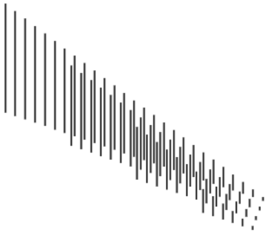
C# COLLECTIONS IN GH

DESCRIPTION **C# SYNTAX**

2D Arrays of Arrays
(Jagged arrays)
are not readable

```
Point3d[][] points = new Point3d[n][];

for (int i = 0; i < n ; i++){
   points[i] = new Point3d[i * i];
      for(int j = 0; j < i * i; j++)
        points[i][j] = new Point3d(  i * dist,  j * dist, 0);
    }

A = points;
```

3D Arrays of Arrays
(Jagged arrays)
are not readable



```
Point3d[][][] points = new Point3d[n1][][];

    for (int i = 0; i < n1 ; i++){
      points[i] = new Point3d[n2][];
      for(int j = 0; j < n2; j++){
        points[i][j] = new Point3d[n3*j];
        for(int k = 0; k < n3*j; k++)
          points[i][j][k] = new Point3d( i * dist,  j * dist,  k * dist);
        }
      }
A = points;
```

DESCRIPTION	C# SYNTAX

2D Multidimensional Arrays (Rectangular Arrays) are readable, but only as a flattened list
 	
	Point3d[,,] points = new Point3d[n, n, n];
	
	for (int i = 0; i < n ; i++)
	for(int j = 0; j < n; j++)
	for(int k = 0; k < n; k++)
	points[i, j, k] = new Point3d(i * dist, j * dist, k * dist);
	
	A = points;

DATATREE

DESCRIPTION	C# SYNTAX
DataTree	DataTree<Point3d> points = new DataTree<Point3d>();
	for (int i = 0; i < n ; i++)
	for(int j = 0; j < n; j++)
	for(int k = 0; k < n; k++)
GH_Path	points.Add(new Point3d(i * dist, j * dist, k * dist), new GH_Path(i, j));
	A = points;

NOISE / RANDOM

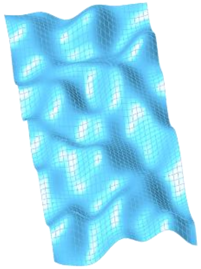
DESCRIPTION **C# SYNTAX**

Random

```
Random random = new Random();
int randomInt = random.Next(-1,5);
double randomDouble = random.NextDouble(0.01, 0.05);
```

Noise

You must reference SimplexNoise.dll



Noise is called here by SimplexNoise. Noise.Generate

```
private void RunScript(double scale, double increment, double zIncrement, int width, int
height, int step, ref object A, ref object B, ref object C, ref object Noise)   {
   int sizeX = (int) (width / step);
   int sizeY = (int) (height / step);
   List<Point3d> points = new List<Point3d>();
   double xOff = 0.0;
   zOff += zIncrement;
   //For every x,y coordinate, calculate a noise value and produce circle radius
   for(int x = 0; x < sizeX * step; x += step){
       xOff += increment;
       double yOff = 0.0;
       for(int y = 0; y < sizeY * step; y += step){
         yOff += increment;
        double dist = SimplexNoise.Noise.Generate(Convert.ToSingle(xOff),
                         Convert.ToSingle(yOff), Convert.ToSingle(zOff)) * scale;
        points.Add(new Point3d(x, y, dist));
     } }
   //Output
   A = points;   B = sizeX;  C = sizeY;
}
double zOff = 0.0; //Output
```

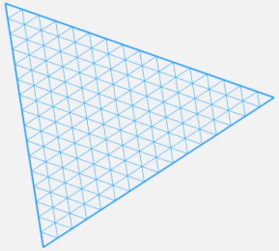
SUBDIVISION

DESCRIPTION C# SYNTAX

Function without recursion.

This is a subdivision example.

```
private void RunScript(int generations, double height, Polyline tri, ref object A)  {
    List < Polyline > a = subdivide(tri, height);
    List < Polyline > b = new List<Polyline>();
    for(int i = 0; i < generations; i++){
        b.Clear();
        for(int j = 0; j < a.Count; j++)
            b.AddRange(subdivide(a[j], height));
        a = new List<Polyline>(b);
    }
    A = a;  //Output
  }

List<Polyline> subdivide(Polyline triangle, double height){
    Line[] segments = triangle.GetSegments();
    Vector3d normal = Vector3d.CrossProduct(segments[0].Direction, segments[1].Direction);
    normal.Unitize();
    normal *= height;
    List<Point3d> midPts = new List<Point3d>(); //Create Middle points
    foreach(Line s in segments){
      Vector3d m = (Vector3d) (s.PointAt(0.5));
      m += normal;
      midPts.Add((Point3d) m);
    }
    Polyline a = new Polyline(new List<Point3d>{segments[0].From, midPts[0], midPts[2],segments[0].From});
    Polyline b = new Polyline(new List<Point3d>{midPts[0], segments[1].From, midPts[1],midPts[0]});
    Polyline c = new Polyline(new List<Point3d>{midPts[2], midPts[1], segments[2].From,midPts[2]});
    Polyline d = new Polyline(new List<Point3d>{midPts[0], midPts[1], midPts[2],midPts[0]});
    return new List<Polyline>{a,b,c,d};
  }
```



RTREE

DESCRIPTION **C# SYNTAX**

Random

```
private void RunScript(List<Line> lineList, double radius, int item, ref object A, ref object B)
 {
  lineList2 = lineList;
  cl.Clear();
  ci.Clear();
```

create rtree

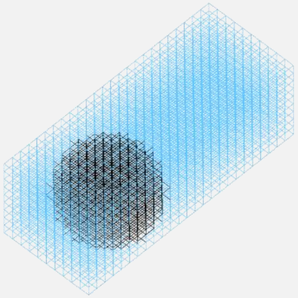
```
  RTree tree = new RTree();
```

add points to rtree

```
  for(int i = 0; i < lineList.Count; i++)
    tree.Insert(lineList[i].PointAt(0.5), i);
```

search

```
  tree.Search(new Sphere(lineList[item].PointAt(0.5), radius), method);
  A = cl;
  B = ci;
 }
```



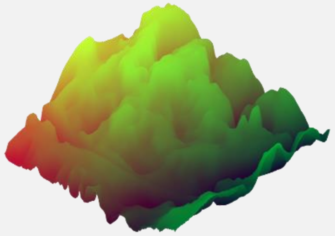
```
 //Custom Additional code
 List<Line> cl = new List<Line>();
 List<int> ci = new List<int>();
 List<Line> lineList2 = new List<Line>();
 private void method(object sender, RTreeEventArgs e){
  cl.Add(lineList2[e.Id]);
  ci.Add(e.Id);
 }
```

BITMAP

DESCRIPTION **C# SYNTAX**

Reference .jpg file

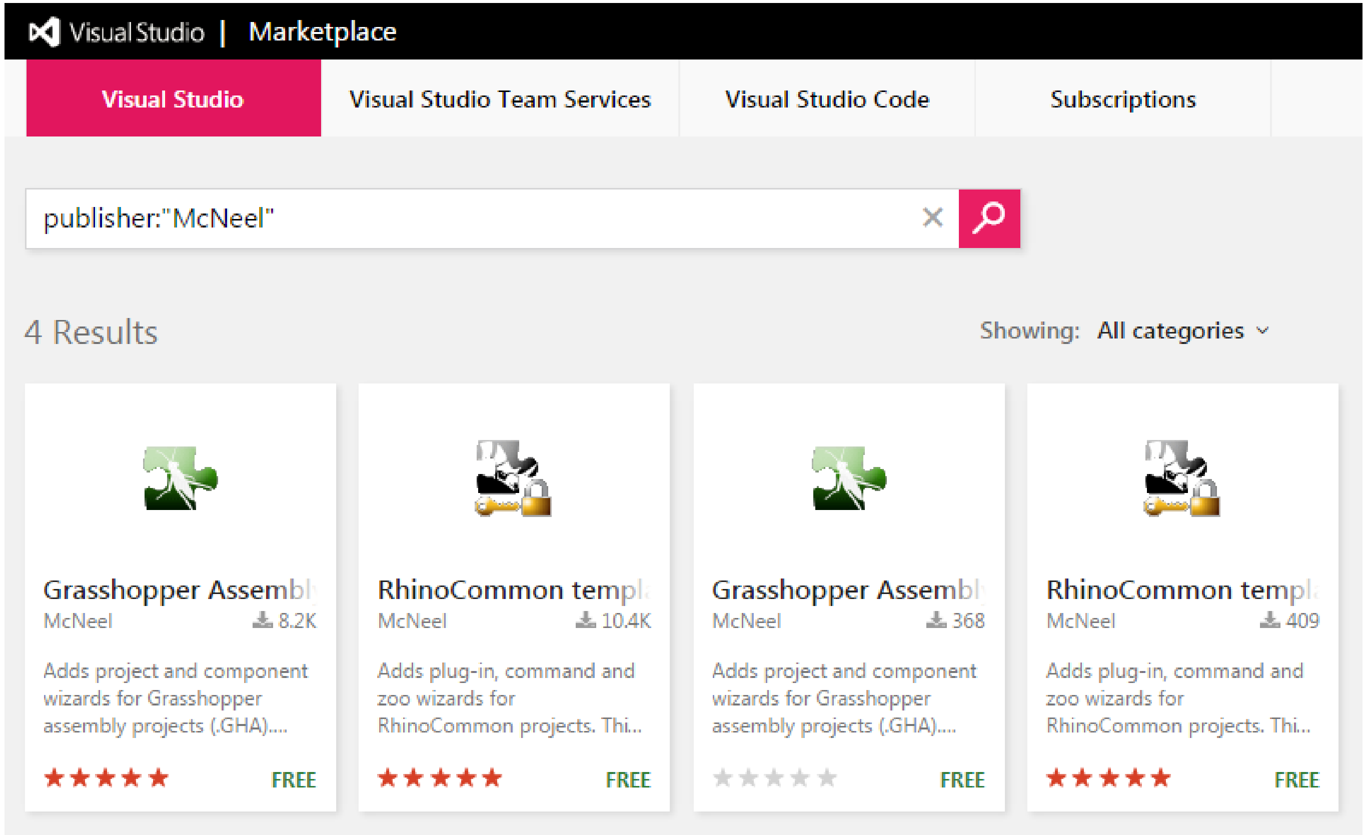
```
if (!System.IO.File.Exists(path))
  throw new ArgumentException("File does not exist");
// Load the bitmap from file.
Bitmap bitmap = new Bitmap(path);
// Resize it to 200x200 pixels.
bitmap = new Bitmap(bitmap, 200, 200);
// Create a mesh and assign height and color values
int nx = bitmap.Width - 1;
int ny = bitmap.Height - 1;
Interval dx = new Interval(0, 200);
Interval dy = new Interval(0, 200);

Mesh mesh = Mesh.CreateFromPlane(Plane.WorldXY, dx, dy, 200, 200);

Color[] colours = new Color[mesh.Vertices.Count];
for (int i = 0; i < mesh.Vertices.Count; i++)
{
  Point3f vertex = mesh.Vertices[i];
  int x = (int) vertex.X;
  int y = (int) vertex.Y;
  colours[i] = bitmap.GetPixel(x, y);
  mesh.Vertices.SetVertex(i, vertex.X, vertex.Y, -colours[i].B * scale);
}
mesh.VertexColors.SetColors(colours);
M = mesh;
```



COMPONENT VISUAL STUDIO

GRASSHOPPER TEMPLATE

DOWNLOAD TEMPLATE



Visual Studio | Marketplace

Visual Studio | Visual Studio Team Services | Visual Studio Code | Subscriptions

publisher:"McNeel"

4 Results

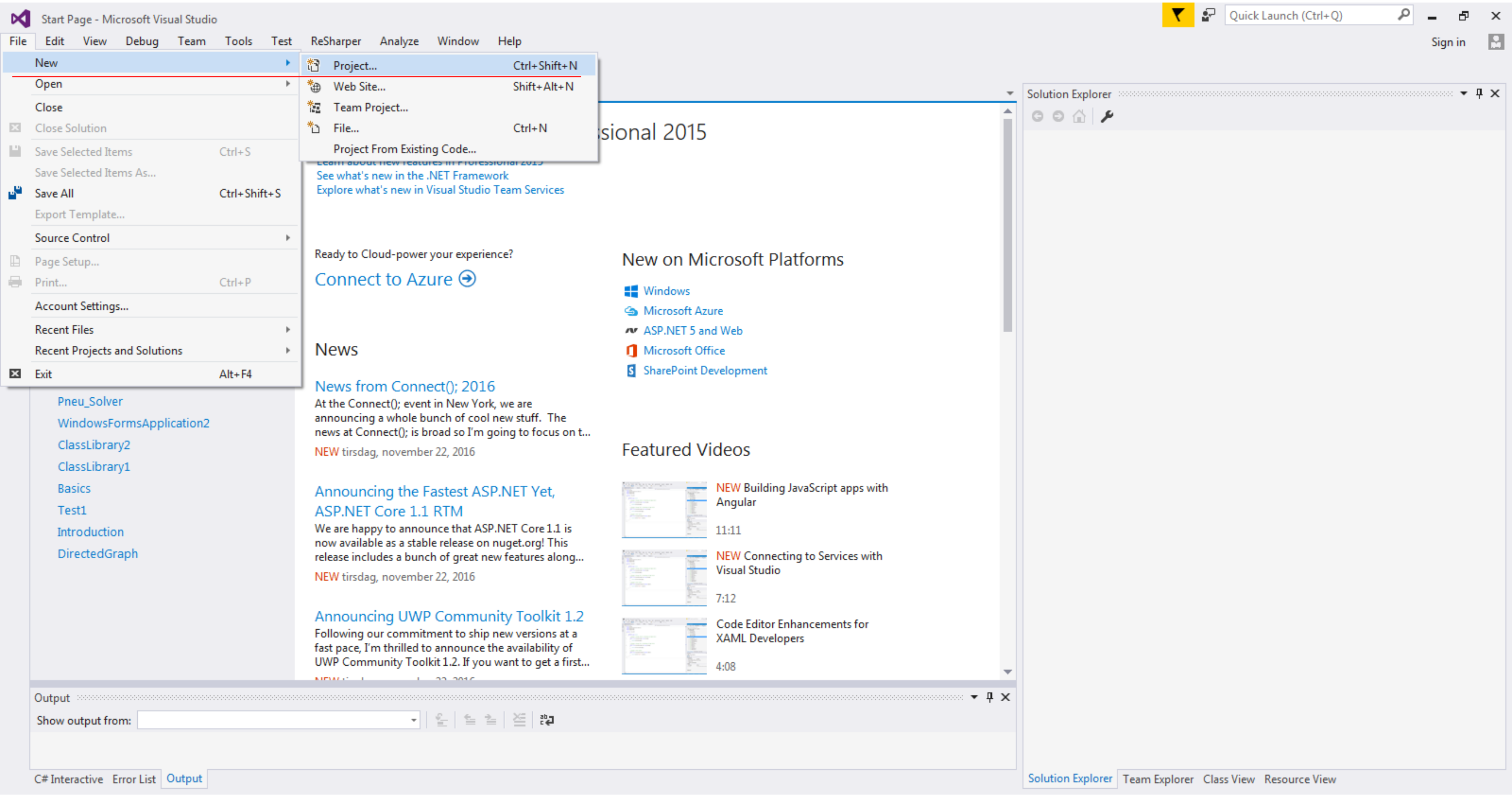
Showing: All categories

Grasshopper Assembl	RhinoCommon templ	Grasshopper Assembl	RhinoCommon templ
McNeel 8.2K	McNeel 10.4K	McNeel 368	McNeel 409
Adds project and component wizards for Grasshopper assembly projects (.GHA)....	Adds plug-in, command and zoo wizards for RhinoCommon projects. Thi...	Adds project and component wizards for Grasshopper assembly projects (.GHA)....	Adds plug-in, command and zoo wizards for RhinoCommon projects. Thi...
★★★★★ FREE	★★★★★ FREE	☆☆☆☆☆ FREE	★★★★★ FREE

https://marketplace.visualstudio.com/items?itemName=McNeel.GrasshopperAssemblyforv5

...AND INSTALL IT (<u>YOU MUST HAVE VISUAL STUDIO</u>)

CREATE NEW PROJECT


Start Page - Microsoft Visual Studio
Quick Launch (Ctrl+Q)
Sign in
File Edit View Debug Team Tools Test ReSharper Analyze Window Help
New
Open
Close
Close Solution
Save Selected Items Ctrl+S
Save Selected Items As...
Save All Ctrl+Shift+S
Export Template...
Source Control
Page Setup...
Print... Ctrl+P
Account Settings...
Recent Files
Recent Projects and Solutions
Exit Alt+F4
Project... Ctrl+Shift+N
Web Site... Shift+Alt+N
Team Project...
File... Ctrl+N
Project From Existing Code...
sional 2015
See what's new in the .NET Framework
Explore what's new in Visual Studio Team Services
Pneu_Solver
WindowsFormsApplication2
ClassLibrary2
ClassLibrary1
Basics
Test1
Introduction
DirectedGraph
Ready to Cloud-power your experience?
Connect to Azure
New on Microsoft Platforms
Windows
Microsoft Azure
ASP.NET 5 and Web
Microsoft Office
SharePoint Development
News
Featured Videos
NEW Building JavaScript apps with Angular
11:11
NEW Connecting to Services with Visual Studio
7:12
Code Editor Enhancements for XAML Developers
4:08
Solution Explorer
Output
Show output from:
C# Interactive Error List Output
Solution Explorer Team Explorer Class View Resource View

News from Connect(); 2016

At the Connect(); event in New York, we are announcing a whole bunch of cool new stuff. The news at Connect(); is broad so I'm going to focus on t...

NEW tirsdag, november 22, 2016

Announcing the Fastest ASP.NET Yet, ASP.NET Core 1.1 RTM

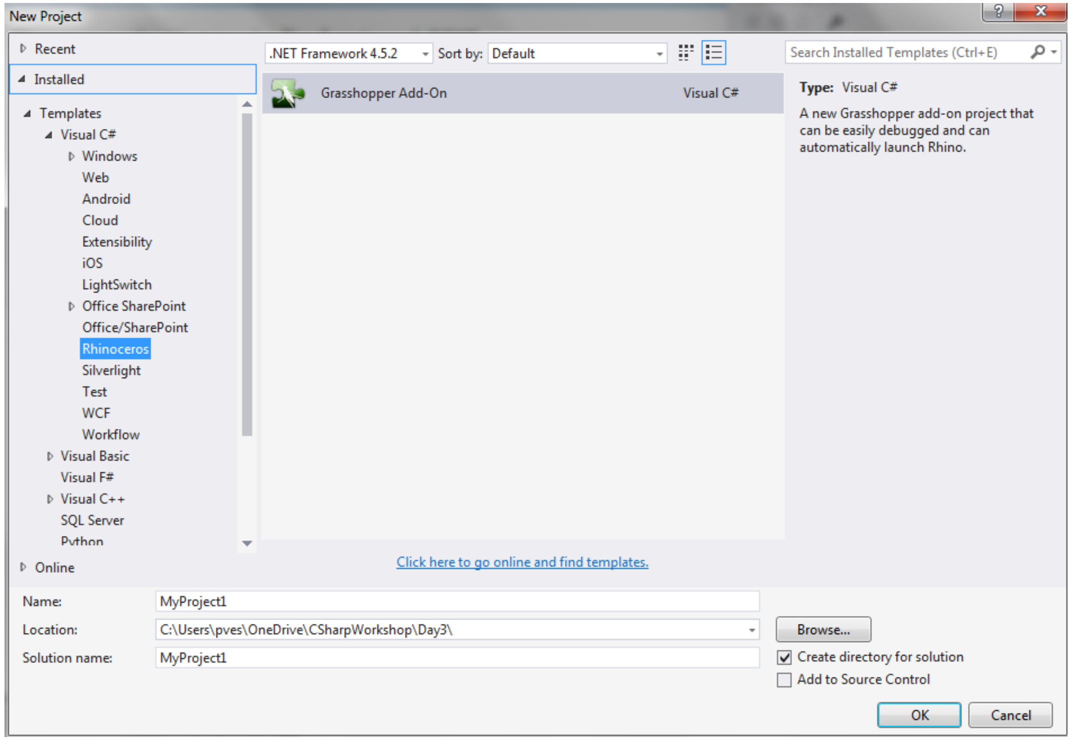
We are happy to announce that ASP.NET Core 1.1 is now available as a stable release on nuget.org! This release includes a bunch of great new features along...

NEW tirsdag, november 22, 2016

Announcing UWP Community Toolkit 1.2

Following our commitment to ship new versions at a fast pace, I'm thrilled to announce the availability of UWP Community Toolkit 1.2. If you want to get a first...

CREATE NEW PROJECT



New Project

.NET Framework 4.5.2 Sort by: Default

Grasshopper Add-On Visual C#

Type: Visual C#

A new Grasshopper add-on project that can be easily debugged and can automatically launch Rhino.

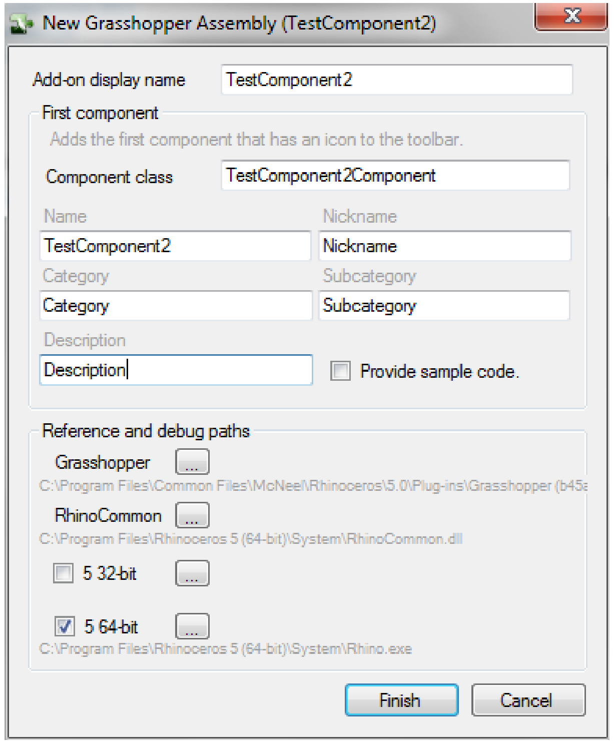
Name: MyProject1

Location: C:\Users\pves\OneDrive\CSharpWorkshop\Day3\

Solution name: MyProject1

☑ Create directory for solution

□ Add to Source Control



New Grasshopper Assembly (TestComponent2)

Add-on display name: TestComponent2

First component

Adds the first component that has an icon to the toolbar.

Component class: TestComponent2Component

Name: TestComponent2

Nickname: Nickname

Category: Category

Subcategory: Subcategory

Description: Description

□ Provide sample code.

Reference and debug paths

Grasshopper
C:\Program Files\Common Files\McNeel\Rhinoceros\5.0\Plug-ins\Grasshopper (b45a

RhinoCommon
C:\Program Files\Rhinoceros 5 (64-bit)\System\RhinoCommon.dll

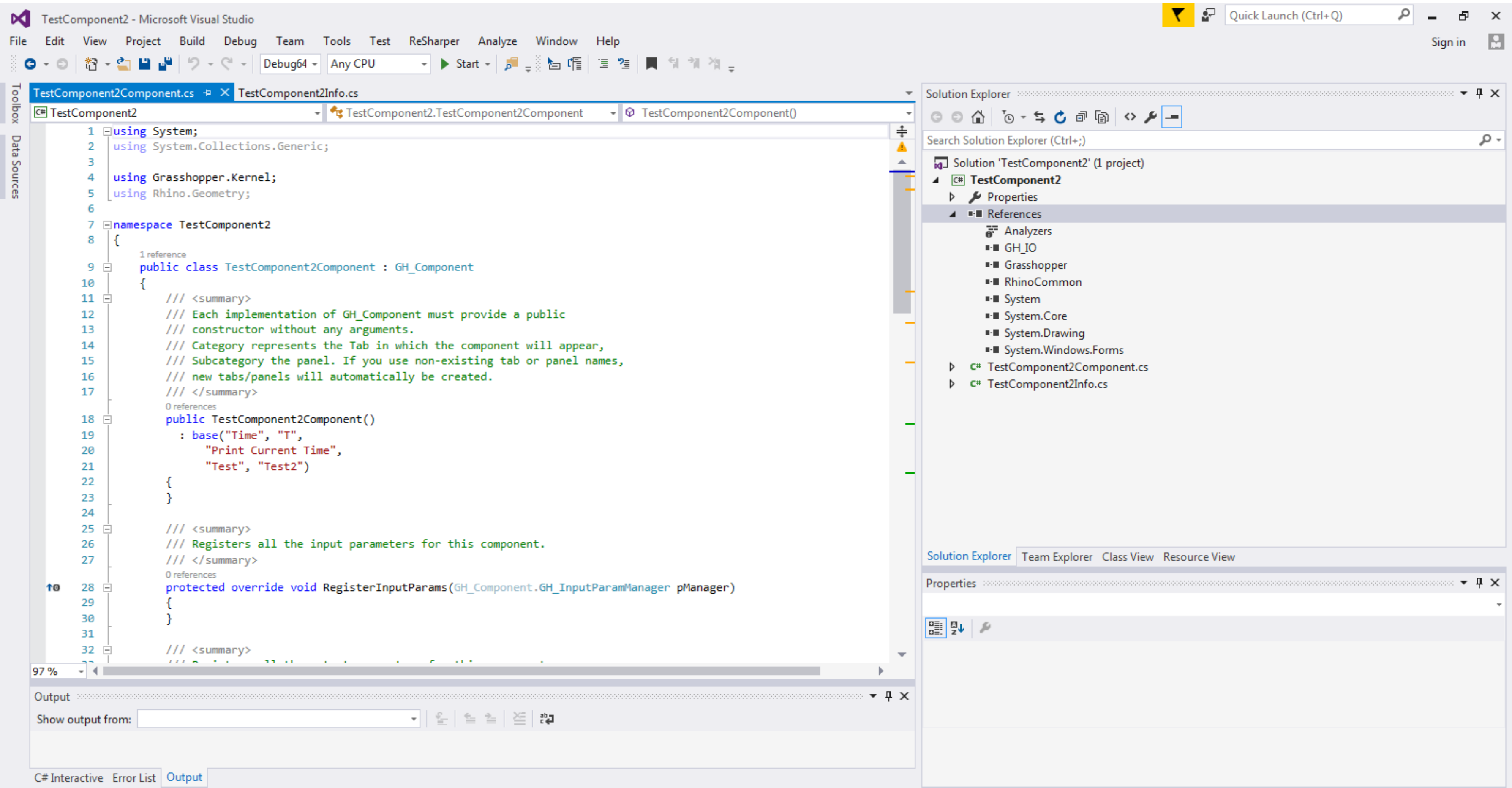
□ 5 32-bit

☑ 5 64-bit
C:\Program Files\Rhinoceros 5 (64-bit)\System\Rhino.exe

1. *Select Grasshopper Add-On template*
   Give your project name and directory

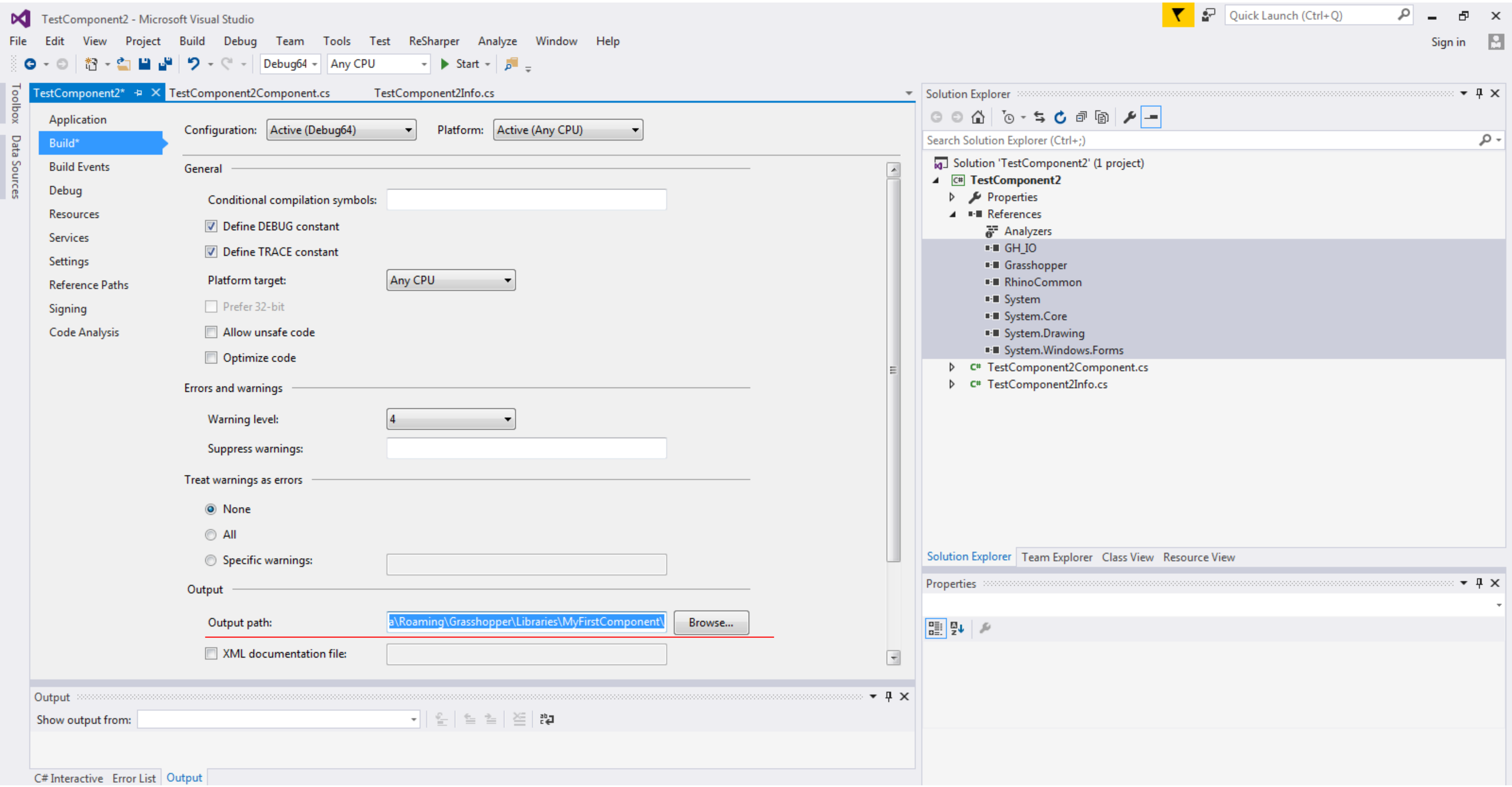
2. *Name you component.*

COMPONENT CLASS



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

using Grasshopper.Kernel;
using Rhino.Geometry;

namespace TestComponent2
{
    1 reference
    public class TestComponent2Component : GH_Component
    {
        /// <summary>
        /// Each implementation of GH_Component must provide a public
        /// constructor without any arguments.
        /// Category represents the Tab in which the component will appear,
        /// Subcategory the panel. If you use non-existing tab or panel names,
        /// new tabs/panels will automatically be created.
        /// </summary>
        0 references
        public TestComponent2Component()
          : base("Time", "T",
              "Print Current Time",
              "Test", "Test2")
        {
        }

        /// <summary>
        /// Registers all the input parameters for this component.
        /// </summary>
        0 references
        protected override void RegisterInputParams(GH_Component.GH_InputParamManager pManager)
        {
        }

        /// <summary>
```

CHANGE BUILD DIRECTORY



Configuration: Active (Debug64)   Platform: Active (Any CPU)

General

Conditional compilation symbols:

☑ Define DEBUG constant

☑ Define TRACE constant

Platform target: Any CPU

□ Prefer 32-bit

□ Allow unsafe code

□ Optimize code

Errors and warnings

Warning level: 4

Suppress warnings:

Treat warnings as errors

● None

○ All

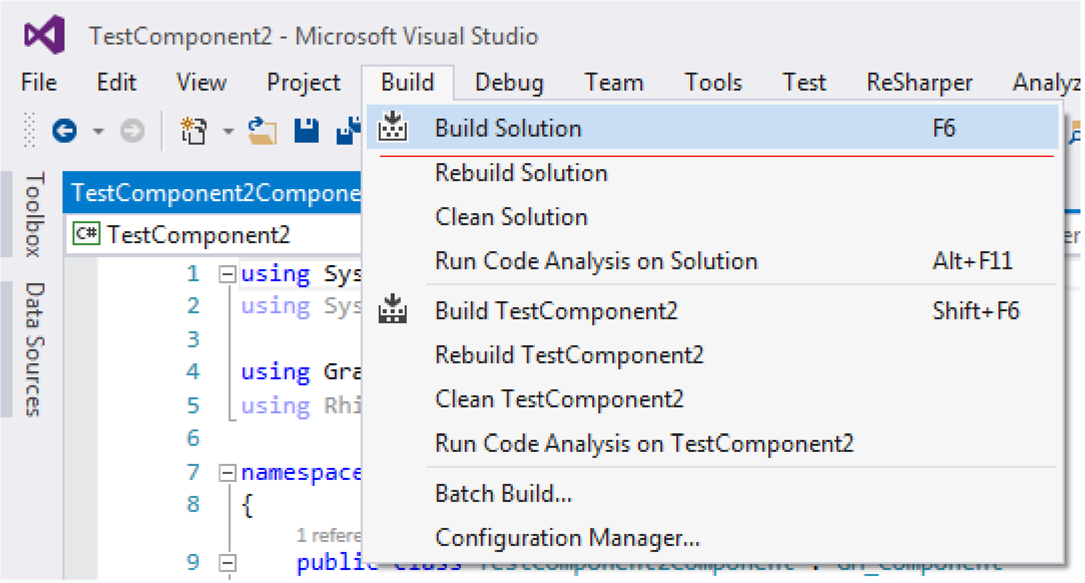
○ Specific warnings:

Output

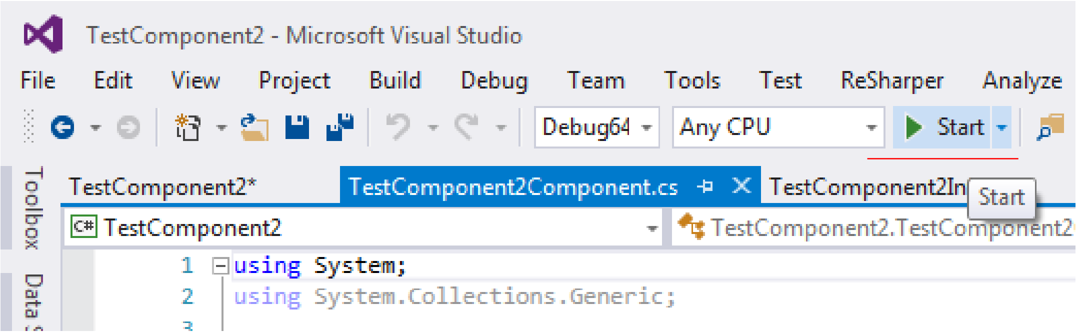
Output path: a\Roaming\Grasshopper\Libraries\MyFirstComponent\ Browse...

□ XML documentation file:

BUILD IT -> RUN IT

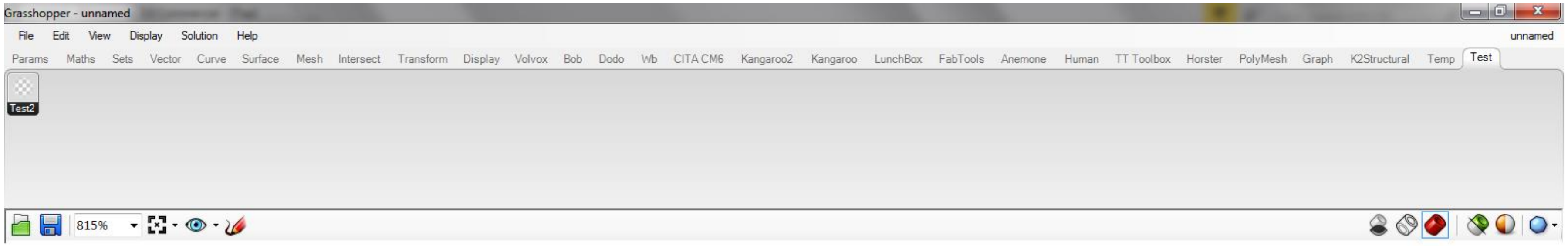


```
using Sys
using Sys

using Gra
using Rhi

namespace
{
    public class TestComponent2Component : GH_Component
```



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

RHINO IS LOADING NOW...



RUN GRASSHOPPER AND FIND YOUR COMPONENT

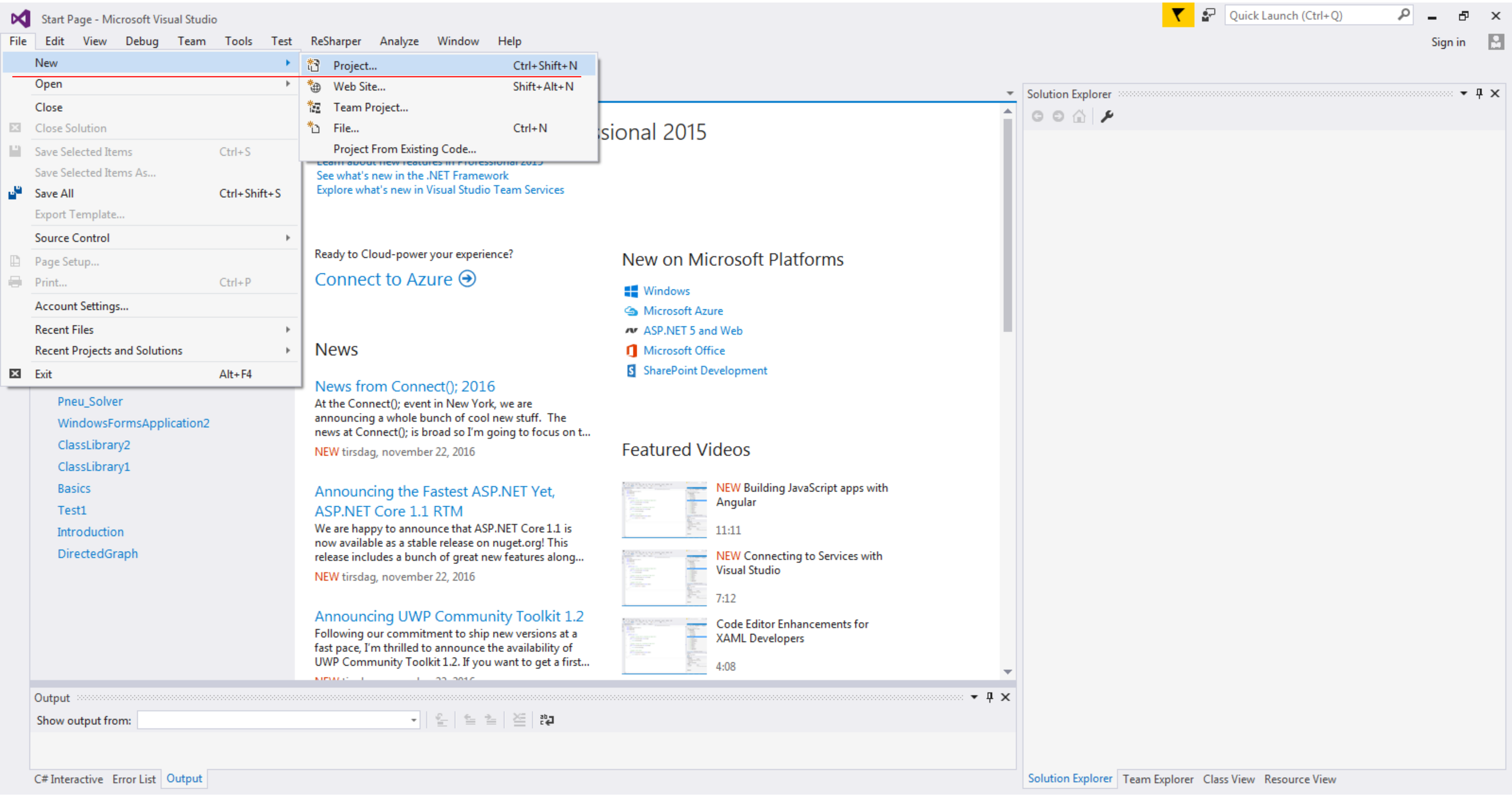
COMPONENT

VISUAL STUDIO

WITHOUT

GRASSHOPPER TEMPLATE

CREATE NEW PROJECT



sional 2015

Learn about new features in Professional 2015

See what's new in the .NET Framework

Explore what's new in Visual Studio Team Services

Ready to Cloud-power your experience?

Connect to Azure

New on Microsoft Platforms

- Windows
- Microsoft Azure
- ASP.NET 5 and Web
- Microsoft Office
- SharePoint Development

News

News from Connect(); 2016

At the Connect(); event in New York, we are announcing a whole bunch of cool new stuff. The news at Connect(); is broad so I'm going to focus on t...

NEW tirsdag, november 22, 2016

Announcing the Fastest ASP.NET Yet, ASP.NET Core 1.1 RTM

We are happy to announce that ASP.NET Core 1.1 is now available as a stable release on nuget.org! This release includes a bunch of great new features along...

NEW tirsdag, november 22, 2016

Announcing UWP Community Toolkit 1.2

Following our commitment to ship new versions at a fast pace, I'm thrilled to announce the availability of UWP Community Toolkit 1.2. If you want to get a first...

Featured Videos

NEW Building JavaScript apps with Angular

11:11

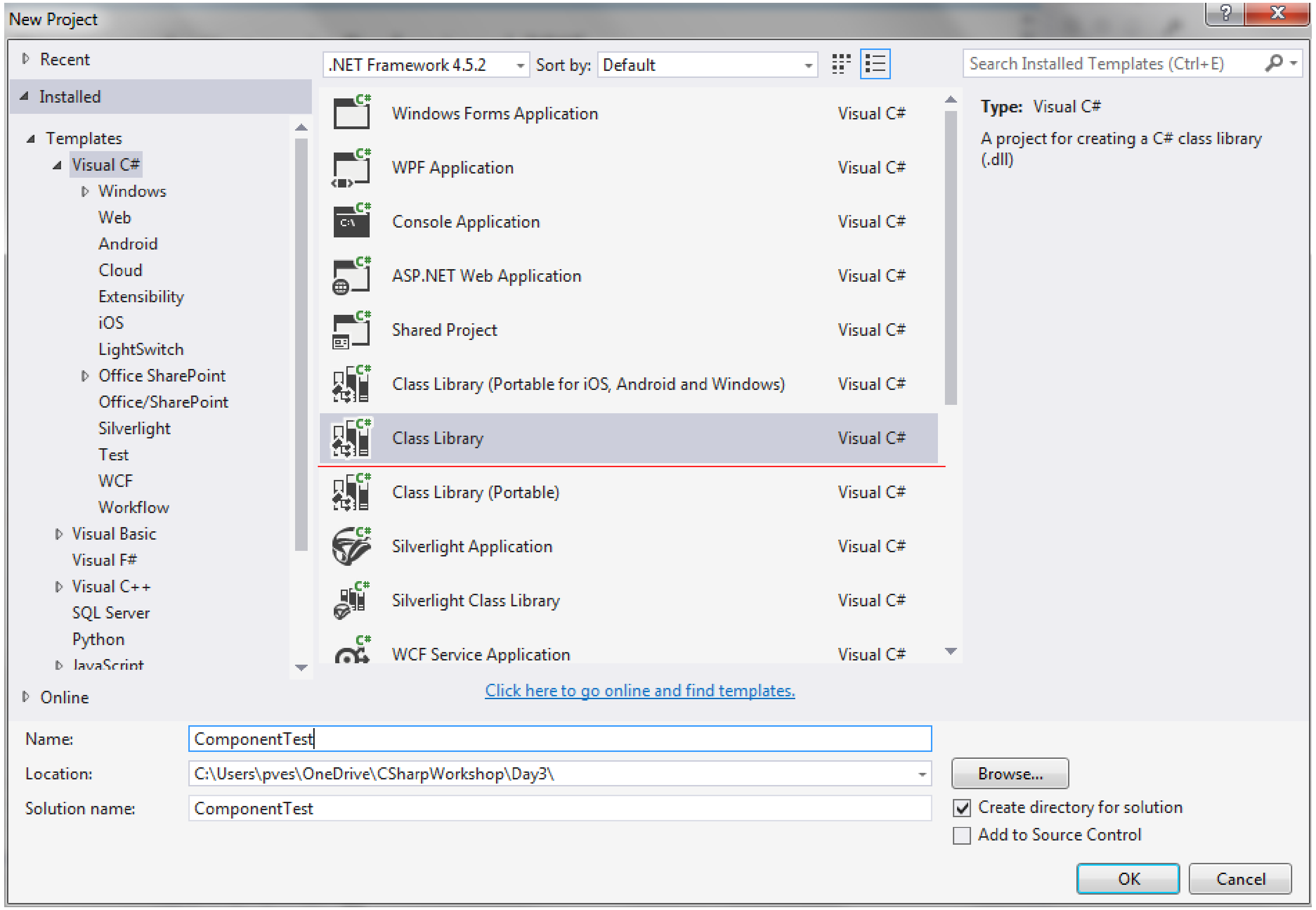
NEW Connecting to Services with Visual Studio

7:12

Code Editor Enhancements for XAML Developers

4:08

CREATE CLASS LIBRARY


New Project
?
Recent
Installed
Templates
Visual C#
Windows
Web
Android
Cloud
Extensibility
iOS
LightSwitch
Office SharePoint
Office/SharePoint
Silverlight
Test
WCF
Workflow
Visual Basic
Visual F#
Visual C++
SQL Server
Python
JavaScript
Online
.NET Framework 4.5.2
Sort by: Default
Search Installed Templates (Ctrl+E)
Windows Forms Application Visual C#
WPF Application Visual C#
Console Application Visual C#
ASP.NET Web Application Visual C#
Shared Project Visual C#
Class Library (Portable for iOS, Android and Windows) Visual C#
Class Library Visual C#
Class Library (Portable) Visual C#
Silverlight Application Visual C#
Silverlight Class Library Visual C#
WCF Service Application Visual C#
Click here to go online and find templates.
Browse...
OK
Cancel

Type: Visual C#

A project for creating a C# class library (.dll)

Name: ComponentTest

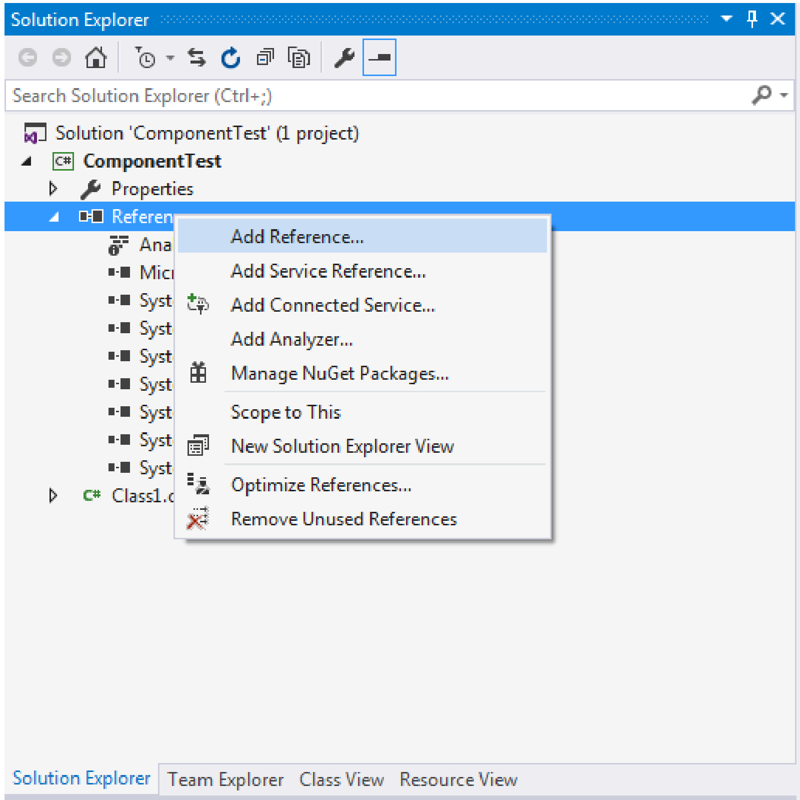
Location: C:\Users\pves\OneDrive\CSharpWorkshop\Day3\

Solution name: ComponentTest

☑ Create directory for solution

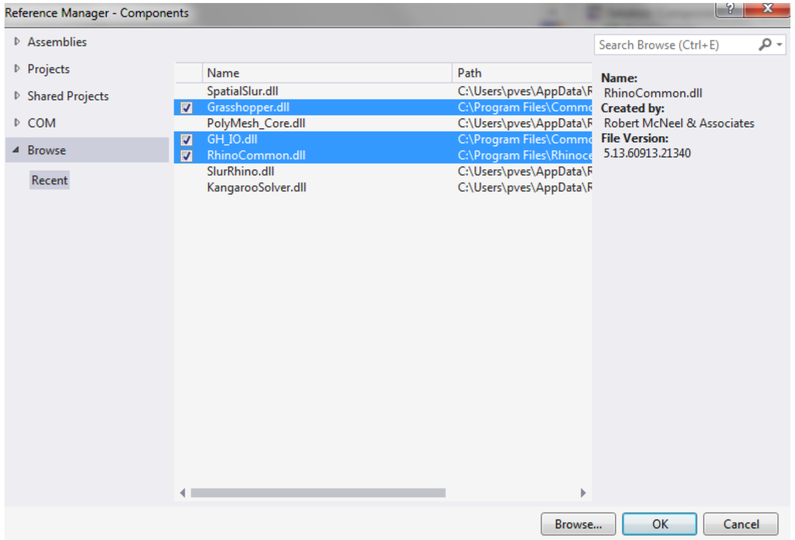
□ Add to Source Control

REFERENCE LIBRARIES



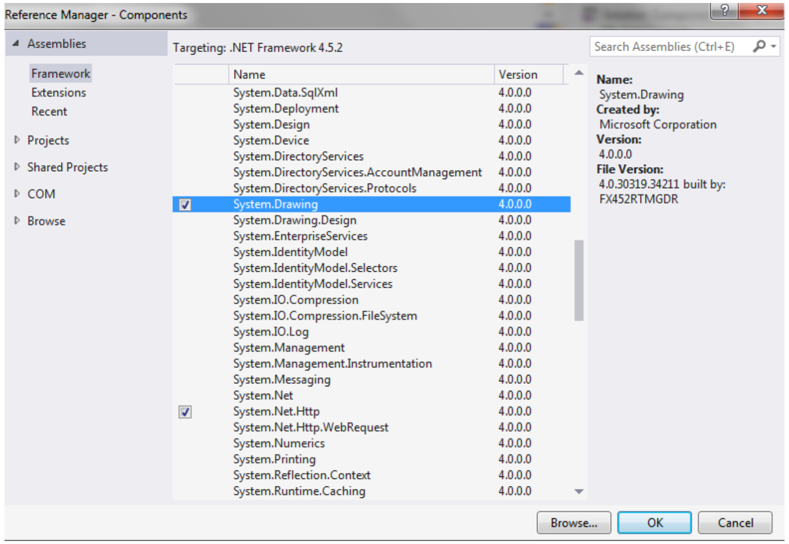
Add 4 references:

1. *System.Drawing*
2. *Grasshopper*
3. *GH_IO*
4. *RhinoCommon*



	Name	Path
	SpatialSlur.dll	C:\Users\pves\AppData\R
☑	Grasshopper.dll	C:\Program Files\Commo
	PolyMesh_Core.dll	C:\Users\pves\AppData\R
☑	GH_IO.dll	C:\Program Files\Commo
☑	RhinoCommon.dll	C:\Program Files\Rhinoce
	SlurRhino.dll	C:\Users\pves\AppData\R
	KangarooSolver.dll	C:\Users\pves\AppData\R

Name:
RhinoCommon.dll
Created by:
Robert McNeel & Associates
File Version:
5.13.60913.21340

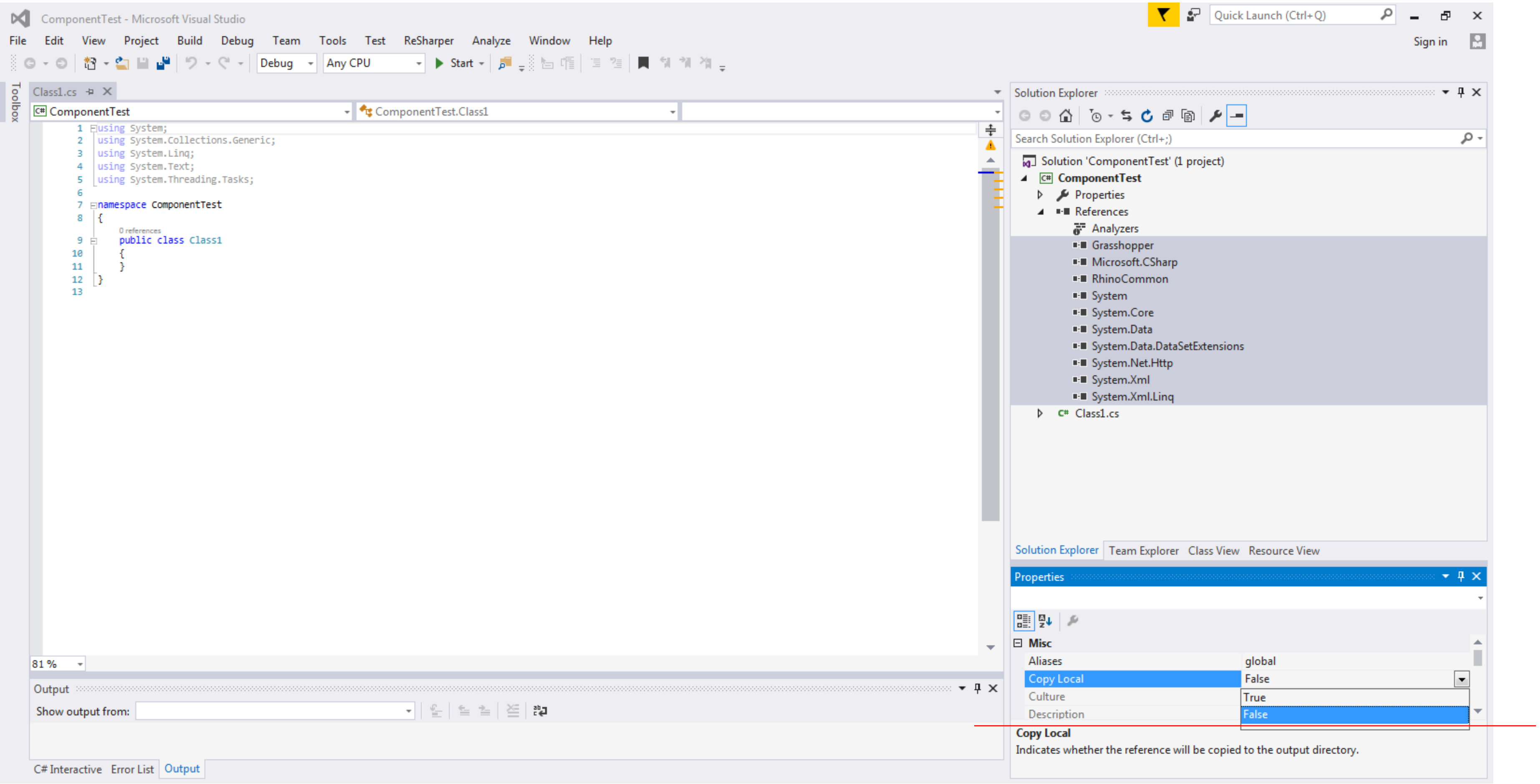


Targeting: .NET Framework 4.5.2

	Name	Version
	System.Data.SqlXml	4.0.0.0
	System.Deployment	4.0.0.0
	System.Design	4.0.0.0
	System.Device	4.0.0.0
	System.DirectoryServices	4.0.0.0
	System.DirectoryServices.AccountManagement	4.0.0.0
	System.DirectoryServices.Protocols	4.0.0.0
☑	System.Drawing	4.0.0.0
	System.Drawing.Design	4.0.0.0
	System.EnterpriseServices	4.0.0.0
	System.IdentityModel	4.0.0.0
	System.IdentityModel.Selectors	4.0.0.0
	System.IdentityModel.Services	4.0.0.0
	System.IO.Compression	4.0.0.0
	System.IO.Compression.FileSystem	4.0.0.0
	System.IO.Log	4.0.0.0
	System.Management	4.0.0.0
	System.Management.Instrumentation	4.0.0.0
	System.Messaging	4.0.0.0
	System.Net	4.0.0.0
☑	System.Net.Http	4.0.0.0
	System.Net.Http.WebRequest	4.0.0.0
	System.Numerics	4.0.0.0
	System.Printing	4.0.0.0
	System.Reflection.Context	4.0.0.0
	System.Runtime.Caching	4.0.0.0

Name:
System.Drawing
Created by:
Microsoft Corporation
Version:
4.0.0.0
File Version:
4.0.30319.34211 built by:
FX452RTMGDR

SET LOCAL COPY TO FALSE


ComponentTest - Microsoft Visual Studio
Quick Launch (Ctrl+Q)
Sign in
File Edit View Project Build Debug Team Tools Test ReSharper Analyze Window Help
Debug
Any CPU
Start
Toolbox
Class1.cs
ComponentTest
ComponentTest.Class1
81 %
Output
Show output from:
C# Interactive Error List Output
Solution Explorer
Search Solution Explorer (Ctrl+;)
Solution Explorer Team Explorer Class View Resource View
Properties

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace ComponentTest
{
    0 references
    public class Class1
    {
    }
}
```

Solution 'ComponentTest' (1 project)
- ComponentTest
  - Properties
  - References
    - Analyzers
    - Grasshopper
    - Microsoft.CSharp
    - RhinoCommon
    - System
    - System.Core
    - System.Data
    - System.Data.DataSetExtensions
    - System.Net.Http
    - System.Xml
    - System.Xml.Linq
  - Class1.cs

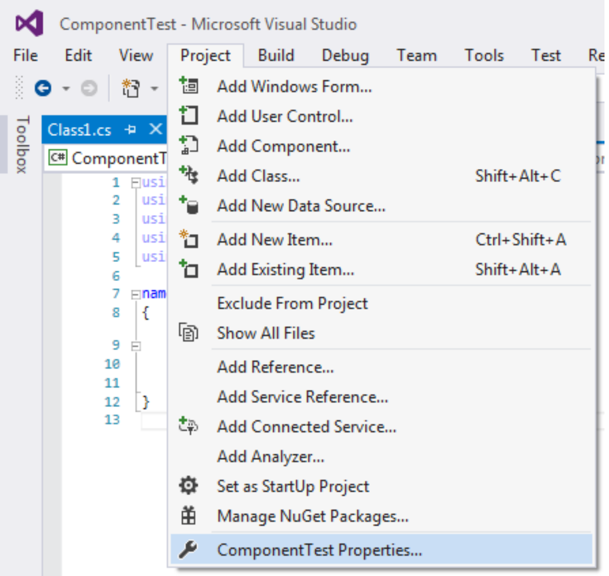
Misc

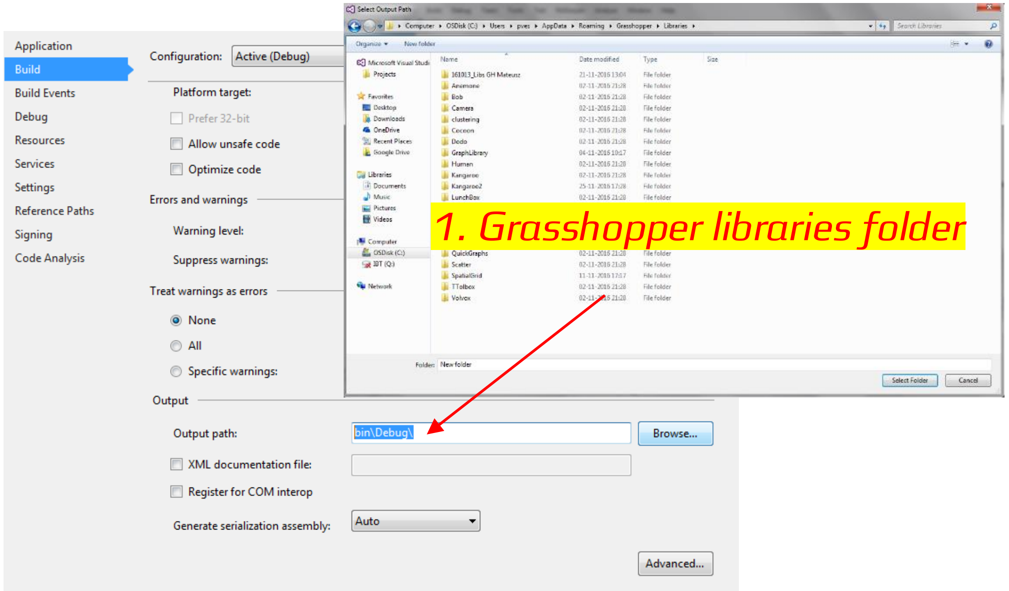
Aliases	global
Copy Local	False
Culture	True
Description	False

Copy Local
Indicates whether the reference will be copied to the output directory.

It prevents copying libraries to your “bin” folder.

CHANGE PROPERTIES





Configuration: Active (Debug)

Platform target:

□ Prefer 32-bit

□ Allow unsafe code

□ Optimize code

Errors and warnings

Warning level:

Suppress warnings:

Treat warnings as errors

● None

○ All

○ Specific warnings:

Output

Output path: bin\Debug\

□ XML documentation file:

□ Register for COM interop

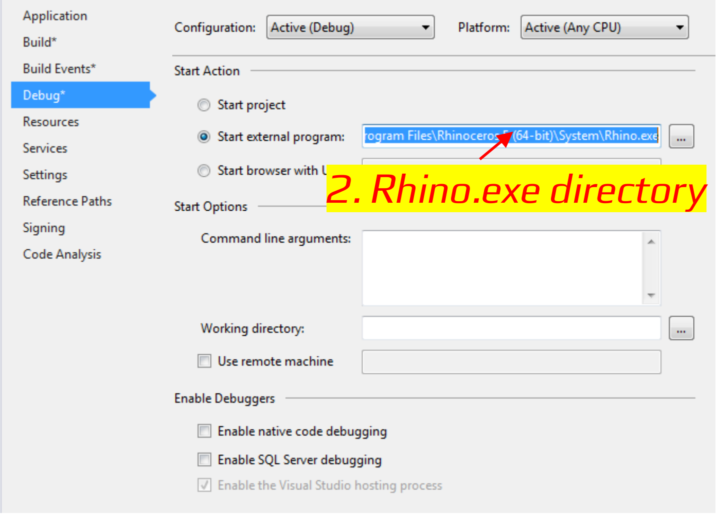
Generate serialization assembly: Auto

Computer ▸ OSDisk (C:) ▸ Users ▸ pves ▸ AppData ▸ Roaming ▸ Grasshopper ▸ Libraries ▸

Name	Date modified	Type	Size
161013_Libs GH Mateusz	21-11-2016 13:04	File folder	
Anemone	02-11-2016 21:28	File folder	
Bob	02-11-2016 21:28	File folder	
Camera	02-11-2016 21:28	File folder	
clustering	02-11-2016 21:28	File folder	
Cocoon	02-11-2016 21:28	File folder	
Dodo	02-11-2016 21:28	File folder	
GraphLibrary	04-11-2016 10:17	File folder	
Human	02-11-2016 21:28	File folder	
Kangaroo	02-11-2016 21:28	File folder	
Kangaroo2	25-11-2016 17:28	File folder	
LunchBox	02-11-2016 21:28	File folder	
QuickGraphs	02-11-2016 21:28	File folder	
Scatter	02-11-2016 21:28	File folder	
SpatialGrid	11-11-2016 17:17	File folder	
TTolbox	02-11-2016 21:28	File folder	
Volvox	02-11-2016 21:28	File folder	

Folder: New folder

1. Grasshopper libraries folder



Configuration: Active (Debug) Platform: Active (Any CPU)

Start Action

○ Start project

● Start external program: rogram Files\Rhinoceros (64-bit)\System\Rhino.exe

○ Start browser with U

Start Options

Command line arguments:

Working directory:

□ Use remote machine

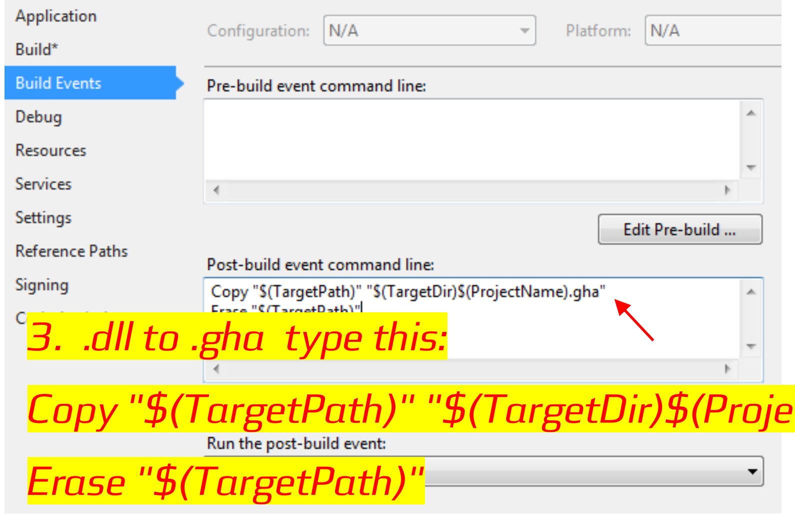
Enable Debuggers

□ Enable native code debugging

□ Enable SQL Server debugging

☑ Enable the Visual Studio hosting process

2. Rhino.exe directory



Configuration: N/A Platform: N/A

Pre-build event command line:

Post-build event command line:

```
Copy "$(TargetPath)" "$(TargetDir)$(ProjectName).gha"
Erase "$(TargetPath)"
```

Run the post-build event:

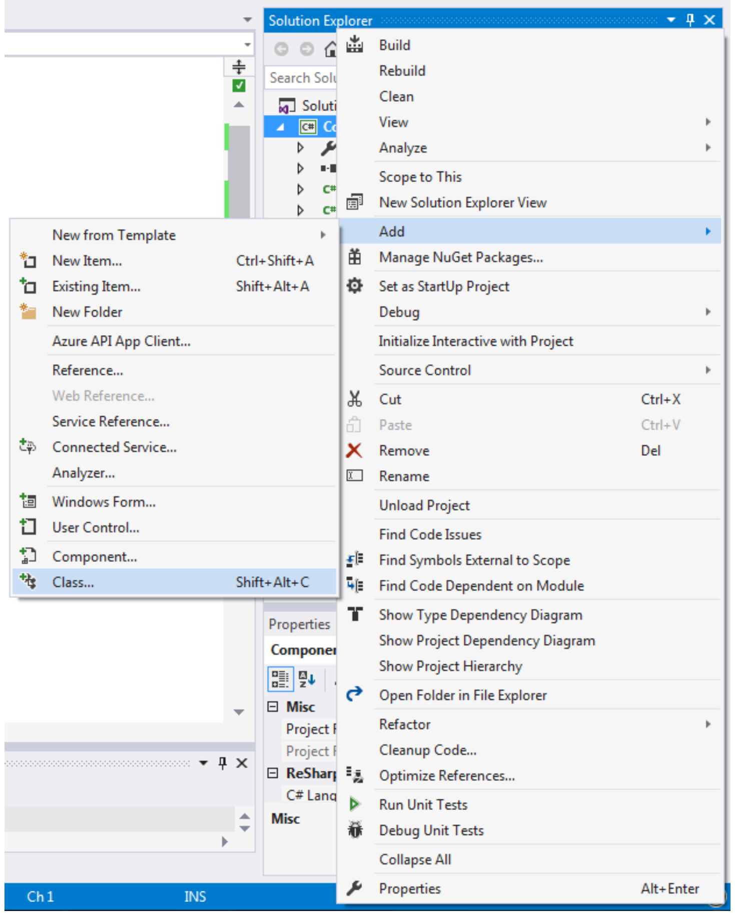
3. .dll to .gha type this:

Copy "$(TargetPath)" "$(TargetDir)$(ProjectName).gha"

Erase "$(TargetPath)"

CREATE ASSEMBLY INFO CLASS

```
using System;
using System.Drawing;
using Grasshopper.Kernel;

namespace Components
{
    public class AssemblyInfo : GH_AssemblyInfo
    {
        public override string Name => "ElephantGun";
        public override Bitmap Icon => null;
        public override string Description => "";
        public override Guid Id => new Guid("35bc357c-5d7d-424f-a91d-a9fe9203cfc8");
        public override string AuthorName => "Petras";
        public override string AuthorContact => "";
    }
}
```

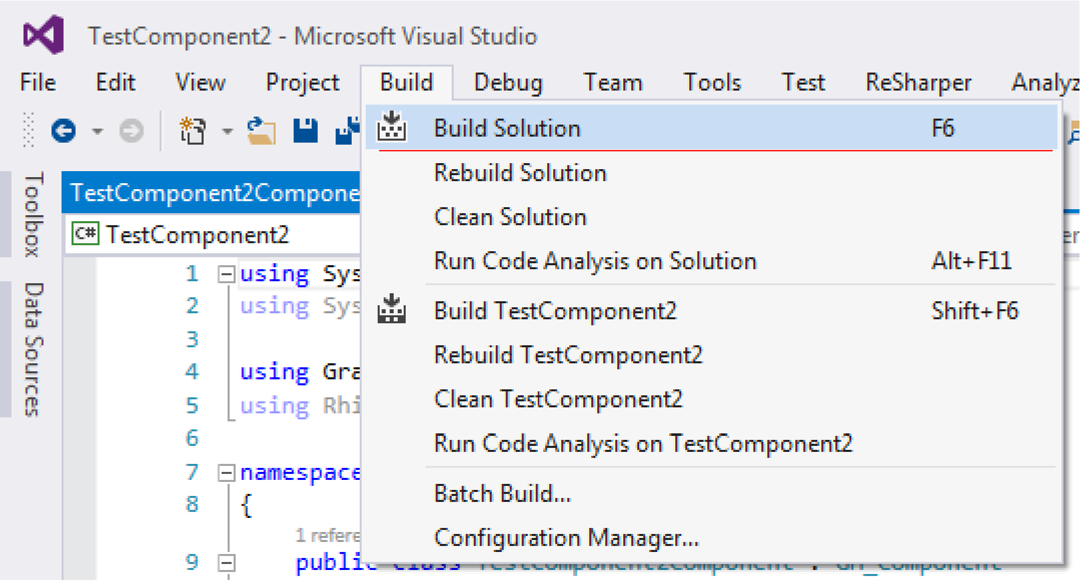


Right click on your project name in solution explorer and create class.
This class is needed for overall assembly not for one component. General Information.

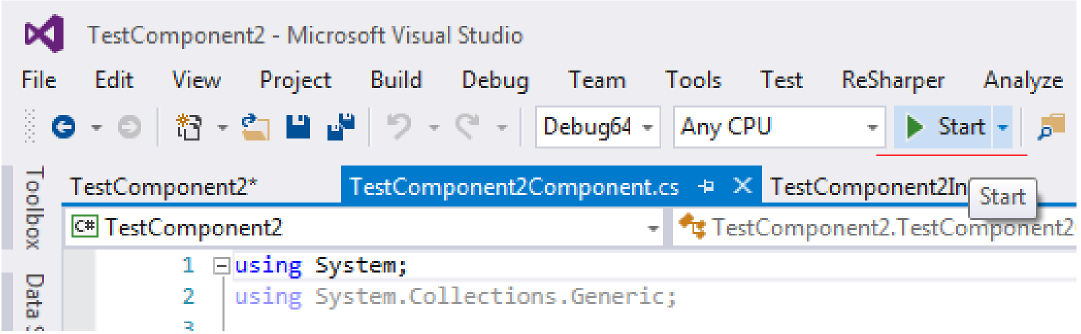
FINALLY WRITE COMPONENT

```
using System;
using System.Collections.Generic;
using Grasshopper.Kernel;
using System.Drawing;

namespace ComponentTest
{
   public class MyFirstComponent : GH_Component
   {
      //Properties
      public override Guid ComponentGuid { get { return new Guid("7e5cbf56-5cf0-4d2b-ada0-4103564afee4"); } }
      protected override Bitmap Icon { get { return null; } }
      //Constructor
      public MyFirstComponent() : base ("MyFirstComponent", "MFC", "Testing my first component", "Temp", "TempSubCategory") { }
      //Methods
      protected override void RegisterInputParams(GH_InputParamManager pManager) {
         pManager.AddNumberParameter("Start", "S", "Start number", GH_ParamAccess.item, 0);
         pManager.AddNumberParameter("Step", "N", "Step number", GH_ParamAccess.item, 0.5);
         pManager.AddIntegerParameter("Count", "C", "How many elements", GH_ParamAccess.item, 10);
      }
      protected override void RegisterOutputParams(GH_OutputParamManager pManager) {
         pManager.AddNumberParameter("Series", "S", "List of numbers", GH_ParamAccess.list);
      }
      protected override void SolveInstance(IGH_DataAccess DA)
      {
         //Get all numbers
         double s = 0.0;
         double n = 0.5;
         int c = 10;
         DA.GetData(0, ref s);
         DA.GetData(1, ref n);
         DA.GetData(2, ref c);
         //Do something
         List<double> numbers = new List<double>();
         for (double i = s; i < n*c; i+=n)
            numbers.Add(i);
         //Output
         DA.SetDataList(0, numbers);
      }
   }
}
```

BUILD IT -> RUN IT

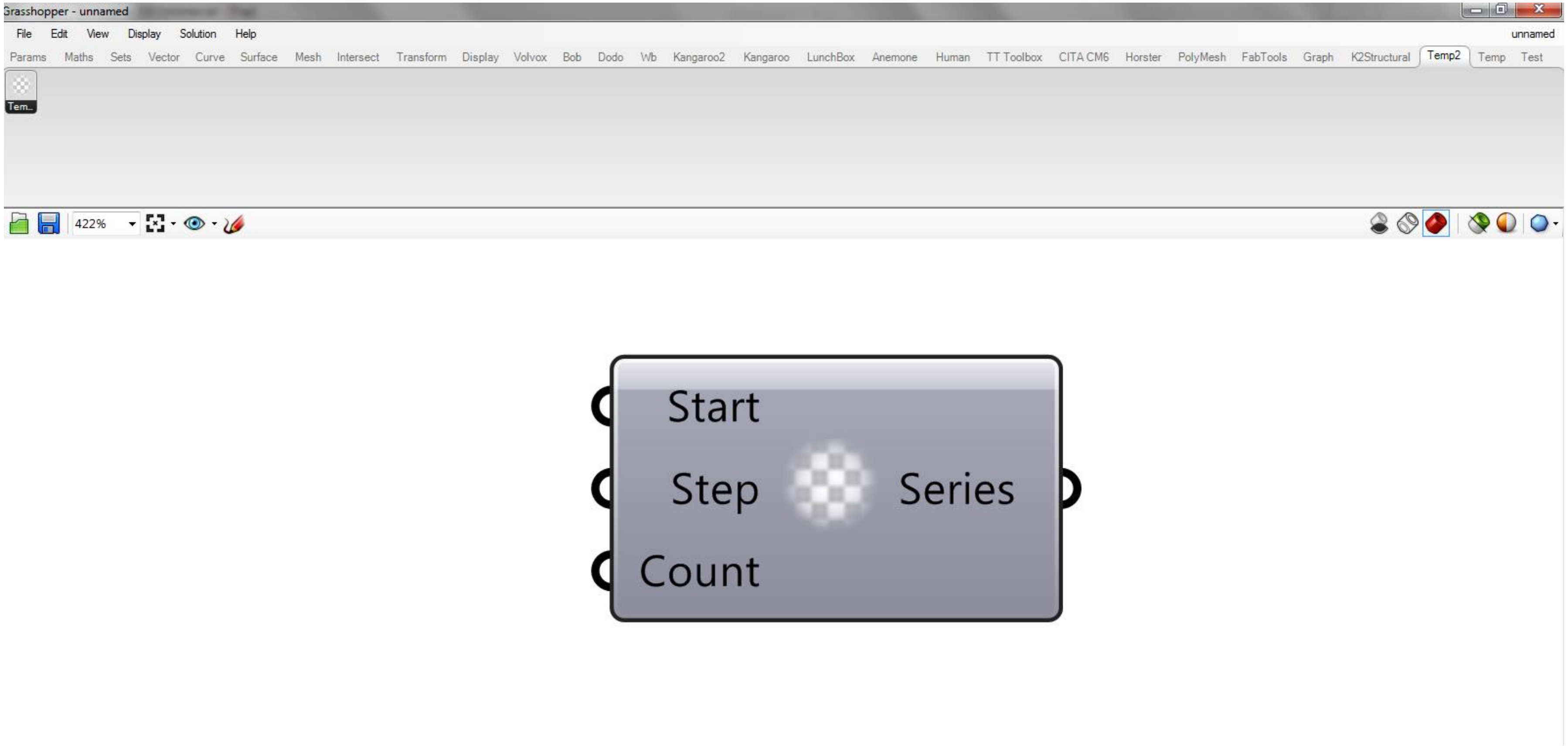


```
using Sys
using Sys

using Gra
using Rhi

namespace
{
    1 refere
    public class TestComponent2Component : GH_Component
```



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

RHINO IS LOADING NOW...


Grasshopper - unnamed
File Edit View Display Solution Help
unnamed
Params Maths Sets Vector Curve Surface Mesh Intersect Transform Display Volvox Bob Dodo Wb Kangaroo2 Kangaroo LunchBox Anemone Human TT Toolbox CITA CM6 Horster PolyMesh FabTools Graph K2Structural Temp2 Temp Test
Tem...
422%
Start
Step
Count
Series

RUN GRASSHOPPER AND FIND YOUR COMPONENT