

Rhino & Grasshopper 入门手册

刘中远 2017/11/1

第二章Grasshopper

- Grasshopper学习目标：
 - 1.掌握gh操作与底层的数据结构、电池/程序的执行优先级。使用gh出视频动画。
 - 2.C#路线：初步掌握C#的语法（参考C++），参考文档，在gh内利用C#调用任意Rhino或gh函数。Python路线：习惯使用python的同学也可以使用python。
 - 3.使用gh内置的C#、python电池调用NativeC++DLL。
 - 4.了解托管与非托管代码。最后可以利用gh调用任意以前自己写好的库或其它开源算法。

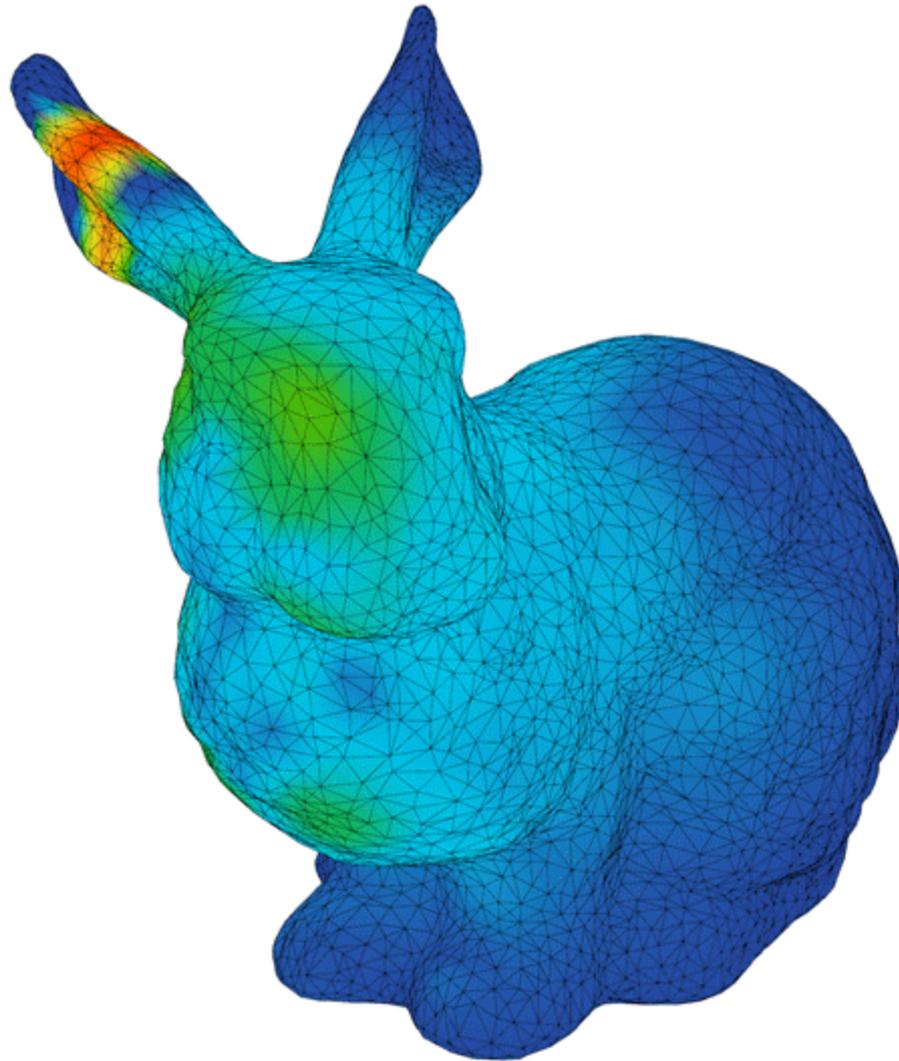
第二章Grasshopper

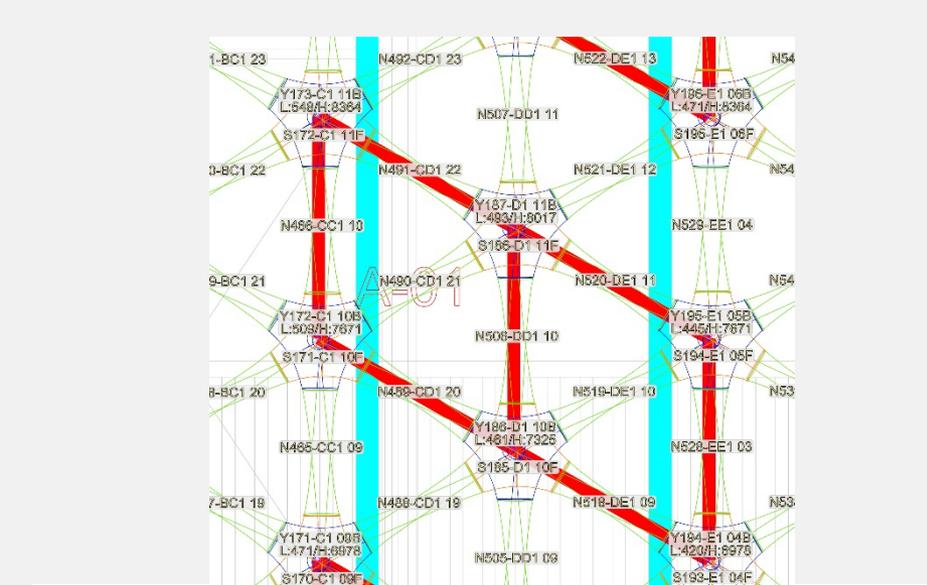
- 2.1 什么是Grasshopper?
- For designers who are exploring new shapes using generative algorithms, Grasshopper® is a graphical algorithm editor tightly integrated with Rhino's 3-D modeling tools. Unlike RhinoScript, Grasshopper requires no knowledge of programming or scripting, but still allows designers to build form generators from the simple to the awe-inspiring.
- 官网<http://www.grasshopper3d.com/>

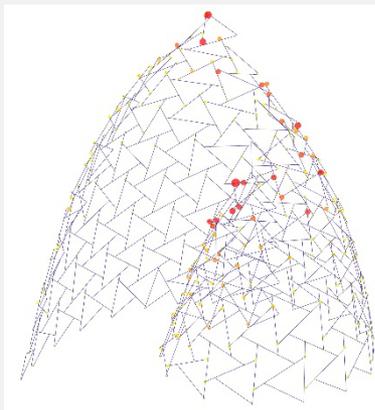
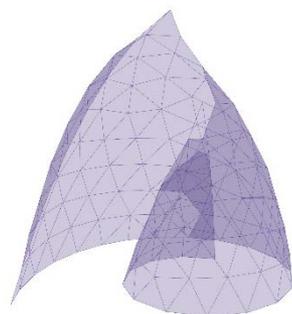
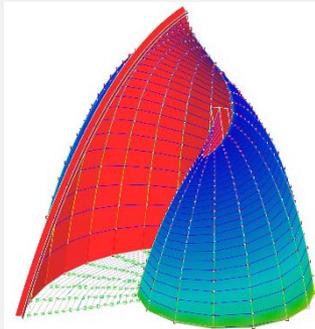
类似节点编程软件：Cycling 74 Max（互动音乐制作平台），unreal的blueprint（游戏开发平台），houdini（特效制作软件），3dmax与maya的材质编辑器，Revit中Dynamo（建筑信息化模型平台）等。

Grasshopper出图案例

四面体剖分着色显示 (gh演示)



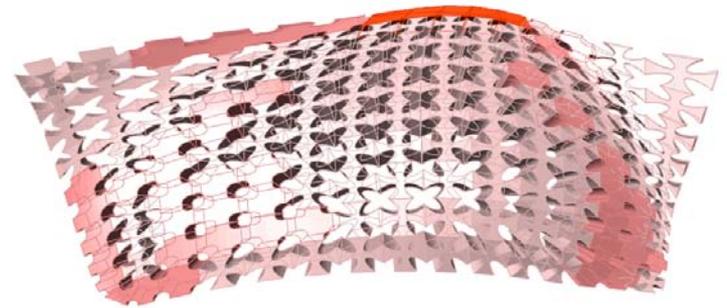
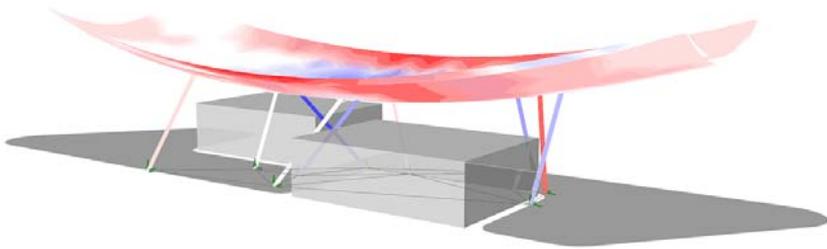
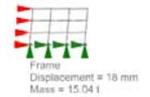
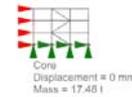
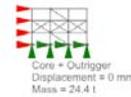
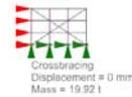
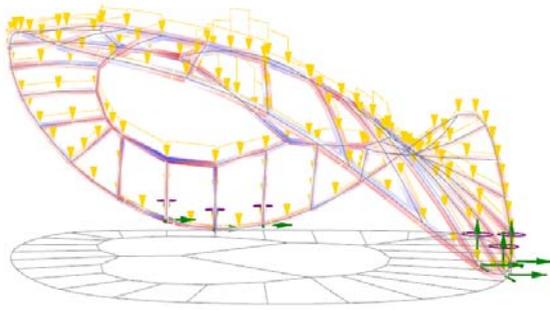






Grasshopper优化案例

Karamba插件（有限元优化）



2.2GH面板介绍

GH面板大体上分3个区域，

文件夹选项

默认电池拖拽区域

gh存放、操作电池的gh画布,详见gh1.jpg

2.3 gh 电池操作

Always:(始终)

RMB+Drag = Pan left/right/up/down 上下左右移动窗口

RMB+Ctrl+Drag = Zoom in/out 缩放窗口

Scroll Wheel = Zoom in/out (注: 缩放中心与鼠标位置有关, 借此可快速远距离移动视窗及电池。)

MMB+Click = Wheel menu 中键菜单

Spacebar = Wheel menu 中键菜单

LMB+Alt+Drag = Split objects (aka 'The Moses Tool')在当前位置横向增加画布面积

While over an empty canvas spot: (当鼠标在空白画布上)

LMB+Drag = selection rectangle 矩形选择

LMB+Shift+Drag = addition selection rectangle 矩形选择加选

LMB+Ctrl+Drag = removal selection rectangle 矩形选择减选

RMB+Click = Canvas menu 画布菜单(功能同中键菜单)

2.3 gh 电池操作

While over an object:(当在一个电池上)

LMB+Drag = Drag selected objects 拖动选择的电池

LMB+Shift+Drag = Drag selected objects along 45-degree guides 垂直移动电池

LMB+Drag+Alt = Copy all dragging objects 复制拖动物体(注: 先拖动再按Alt)

LMB+Click = Either select one object or do nothing if object is already selected
选择

LMB+Shift+Click = Add an object to the selection 加选

LMB+Ctrl+Click = Remove an object from the selection 减选

LMB+Ctrl+Shift+Click = Toggle the selection state 切换选择状态

LMB+Ctrl+Alt = Info mode 显示电池在菜单的位置

LMB+Ctrl+Shift+Alt = Save info mode as image 保存位置信息图片

RMB+Click = Object context menu 电池详细菜单

2.3 gh 电池操作

While over an input/output grip: (当鼠标在一个输入/输出端上)

LMB+Drag = Create a new wire 创建一根新的线, 替换原来的所有连线

LMB+Shift+Drag = Create a new wire without erasing old wires 链接一根新的线, 不影响已经链接上的

LMB+Ctrl+Drag = Erase an existing wire by tracing over it 擦掉已经链接的线

LMB+Ctrl+Shift+Drag = Move all existing wires to some other grip 把所有链接的线移到其它电池上, 不会影响其它电池已有的连线

LMB+Drag+RMB = Create a new wire without exiting the wire tool 连续创建新的链接状态 替换

LMB+Drag+RMB+Shift = Add a wire without exiting the wire tool 连续创建新的链接状态 增加

LMB+Drag+RMB+Control = Remove a wire without exiting the wire tool 连续创建新的链接状态 删除

While over a component icon on the toolpanels: (当电池在工具面板里)

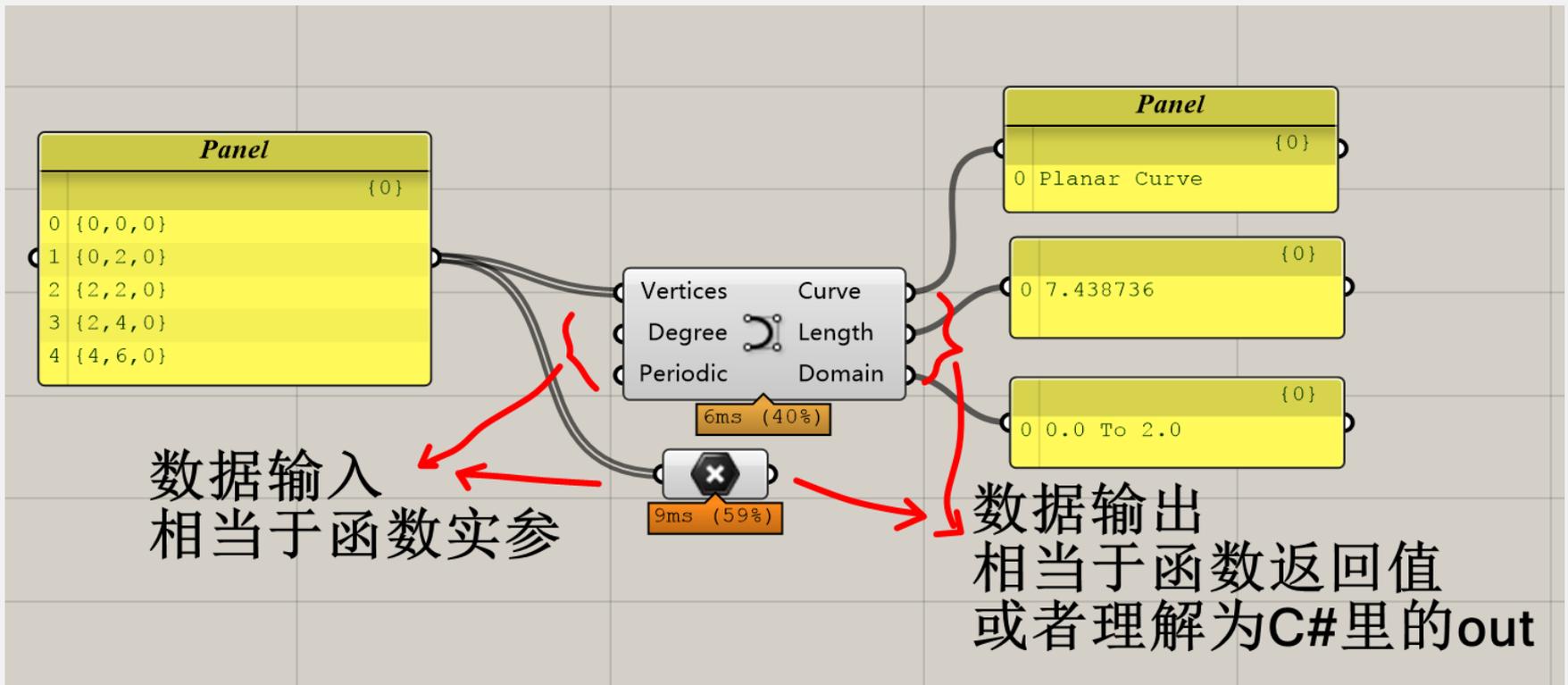
LMB+Shift+Click = Create a component aggregate 一次抓取多个电池

2.4 gh运行原理

GH电池相当于一个函数，函数输入-计算-输出的过程是实时的

左侧是输入端，右侧是输出端，不使用插件的情况下，输出后的结果不可以再连回输入。

因此不支持类似for(){}或者do{}while();的循环计算，但可以通过数据结构处理完成类似需求。



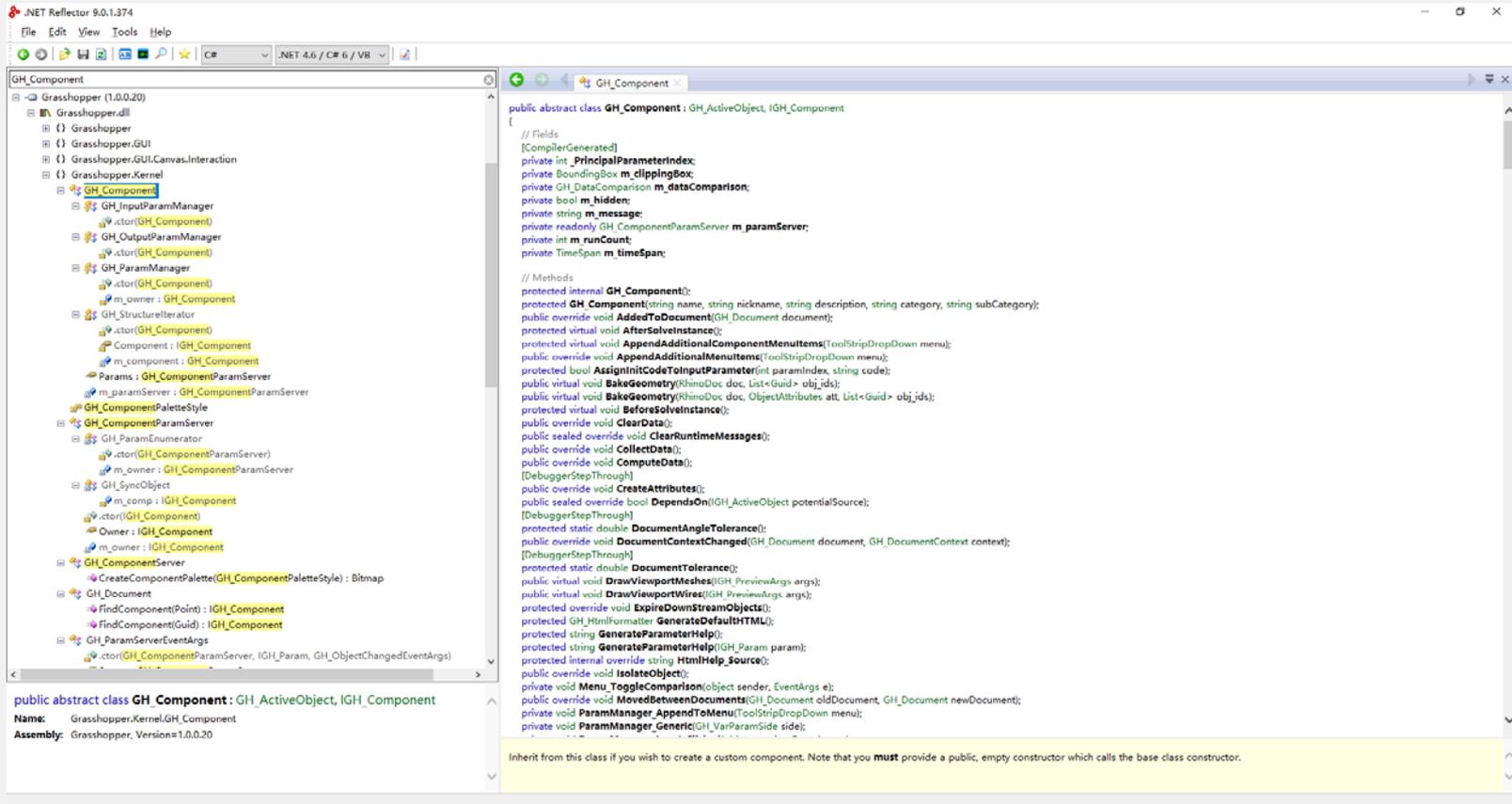
2.5 电池反编译

对grasshopper.dll进行反编译，可以看到所有电池对Rhino命令的调用关系

比如使用.NET Reflector 9.0对其反编译的结果（下一页）：

其中Grasshopper.Kernel.GH_Component类是所有电池的父类，它定义了所有电池的接口（输入输出的类型、个数）、gh在画布上的显示的名称以及图标等

推荐免费的反编译软件：<https://github.com/Oxd4d/dnSpy>，中文版详见课程资料



The screenshot displays the .NET Reflector 9.0.1.374 interface. On the left, a tree view shows the assembly structure for Grasshopper (1.0.0.20), with the `GH_Component` class highlighted. The right pane shows the source code for `GH_Component`, which is an abstract class inheriting from `GH_ActiveObject` and `IGH_Component`. The code includes fields for parameters, clipping boxes, and messages, and defines various methods for document management, menu handling, and data processing. A note at the bottom states: "Inherit from this class if you wish to create a custom component. Note that you **must** provide a public, empty constructor which calls the base class constructor."

```
public abstract class GH_Component : GH_ActiveObject, IGH_Component
{
    // Fields
    [CompilerGenerated]
    private int _principalParameterIndex;
    private BoundingBox m_clippingBox;
    private GH_DataComparison m_dataComparison;
    private bool m_hidden;
    private string m_message;
    private readonly GH_ComponentParamServer m_paramServer;
    private int m_runCount;
    private TimeSpan m_timeSpan;

    // Methods
    protected internal GH_Component();
    protected GH_Component(string name, string nickname, string description, string category, string subCategory);
    public override void AddedToDocument(GH_Document document);
    protected virtual void AfterSolveInstance();
    protected virtual void AppendAdditionalComponentMenuItems(ToolStripDropDown menu);
    public override void AppendAdditionalMenuItems(ToolStripDropDown menu);
    protected bool AssignInitCodeToInputParameter(int parameterIndex, string code);
    public virtual void BakeGeometry(RhinoDoc doc, List<Guid> obj_ids);
    public virtual void BakeGeometry(RhinoDoc doc, ObjectAttributes att, List<Guid> obj_ids);
    protected virtual void BeforeSolveInstance();
    public override void ClearData();
    public sealed override void ClearRuntimeMessages();
    public override void CollectData();
    public override void ComputeData();
    [DebuggerStepThrough]
    public override void CreateAttributes();
    public sealed override bool DependsOn(IGH_ActiveObject potentialSource);
    [DebuggerStepThrough]
    protected static double DocumentAngleTolerance();
    public override void DocumentContextChanged(GH_Document document, GH_DocumentContext context);
    [DebuggerStepThrough]
    protected static double DocumentTolerance();
    public virtual void DrawViewportMeshes(GH_PreviewArgs args);
    public virtual void DrawViewportWires(IGH_PreviewArgs args);
    protected override void ExpireDownStreamObjects();
    protected GH_HtmlFormatter GenerateDefaultHTML();
    protected string GenerateParameterHelp();
    protected string GenerateParameterHelp(IGH_Param param);
    protected internal override string HtmlHelp_Source();
    public override void IsolateObject();
    private void Menu_ToggleComparison(object sender, EventArgs e);
    public override void MovedBetweenDocuments(GH_Document oldDocument, GH_Document newDocument);
    private void ParamManager_AppendToMenu(ToolStripDropDown menu);
    private void ParamManager_Generic(GH_VarParamSide side);

    Inherit from this class if you wish to create a custom component. Note that you must provide a public, empty constructor which calls the base class constructor.
}
```

2.5 电池反编译

所有具体电池的实现都在一些特殊的dll里，这些dll改了名字叫.gha

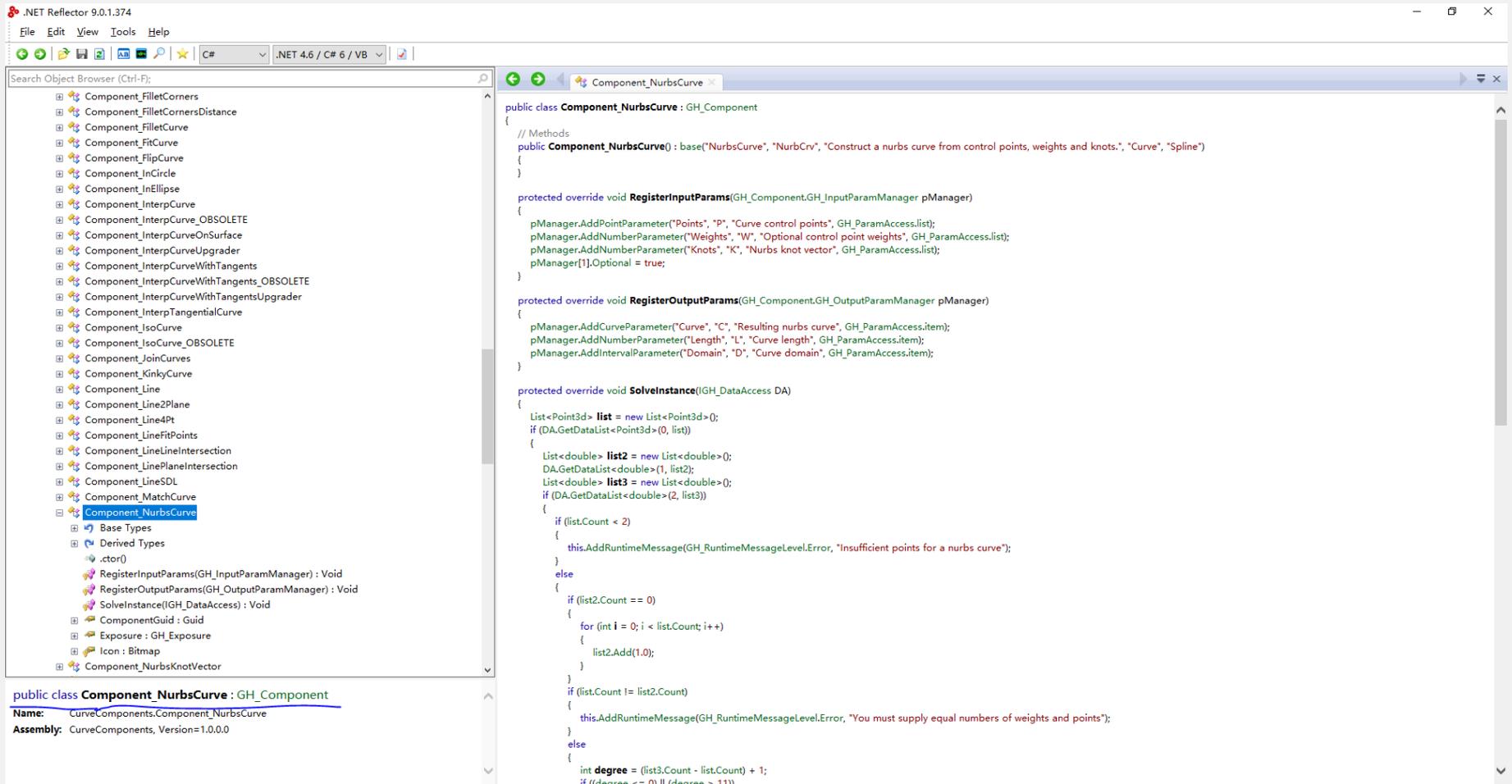
.gha本身也是dll，比如找到我们刚才创建NURBS曲线的例子中使用的电池所在的gha（dll）

C:\Program Files\Common Files\McNeel\Rhinoceros\5.0\Plug-ins\Grasshopper (b45a29b1-4343-4035-989e-044e8580d9cf)\0.9.76.0\Components

 Curve.gha	2016/7/23 20:08	Grasshopper As...	706 KB
 Field.gha	2016/7/23 20:08	Grasshopper As...	40 KB
 Galapagos.dll	2016/7/23 20:08	应用程序扩展	772 KB
 GalapagosLibrary.gha	2016/7/23 20:08	Grasshopper As...	185 KB
 IOLibrary.gha	2016/7/23 20:08	Grasshopper As...	56 KB
 LegacyScript.gha	2016/7/23 20:08	Grasshopper As...	87 KB
 Mathematics.gha	2016/7/23 20:08	Grasshopper As...	621 KB
 Script.gha	2016/7/23 20:08	Grasshopper As...	114 KB
 Surface.gha	2016/7/23 20:08	Grasshopper As...	454 KB
 Transform.gha	2016/7/23 20:08	Grasshopper As...	196 KB
 Triangulation.gha	2016/7/23 20:08	Grasshopper As...	107 KB
 Vector.gha	2016/7/23 20:08	Grasshopper As...	263 KB

2.5 电池反编译

对Curve.gha进行反编译,找到Component_NurbsCurve类,
这个类就是这个Nurbs Curve电池,反编译可以看见内部的函数实现



```
public class Component_NurbsCurve : GH_Component
{
    // Methods
    public Component_NurbsCurve() : base("NurbsCurve", "NurbCrv", "Construct a nurbs curve from control points, weights and knots.", "Curve", "Spline")
    {
    }

    protected override void RegisterInputParams(GH_Component.GH_InputParamManager pManager)
    {
        pManager.AddPointParameter("Points", "P", "Curve control points", GH_ParamAccess.list);
        pManager.AddNumberParameter("Weights", "W", "Optional control point weights", GH_ParamAccess.list);
        pManager.AddNumberParameter("Knots", "K", "Nurbs knot vector", GH_ParamAccess.list);
        pManager[1].Optional = true;
    }

    protected override void RegisterOutputParams(GH_Component.GH_OutputParamManager pManager)
    {
        pManager.AddCurveParameter("Curve", "C", "Resulting nurbs curve", GH_ParamAccess.item);
        pManager.AddNumberParameter("Length", "L", "Curve length", GH_ParamAccess.item);
        pManager.AddIntervalParameter("Domain", "D", "Curve domain", GH_ParamAccess.item);
    }

    protected override void SolveInstance(IGH_DataAccess DA)
    {
        List<Point3d> list = new List<Point3d>(0);
        if (DA.GetDataList<Point3d>(0, list))
        {
            List<double> list2 = new List<double>(0);
            DA.GetDataList<double>(1, list2);
            List<double> list3 = new List<double>(0);
            if (DA.GetDataList<double>(2, list3))
            {
                if (list.Count < 2)
                {
                    this.AddRuntimeMessage(GH_RuntimeMessageLevel.Error, "Insufficient points for a nurbs curve");
                }
                else
                {
                    if (list2.Count == 0)
                    {
                        for (int i = 0; i < list.Count; i++)
                        {
                            list2.Add(1.0);
                        }
                    }
                    if (list.Count != list2.Count)
                    {
                        this.AddRuntimeMessage(GH_RuntimeMessageLevel.Error, "You must supply equal numbers of weights and points");
                    }
                    else
                    {
                        int degree = (list3.Count - list.Count) + 1;
                        if ((degree <= 0) || (degree > 11))
                    }
                }
            }
        }
    }
}
```

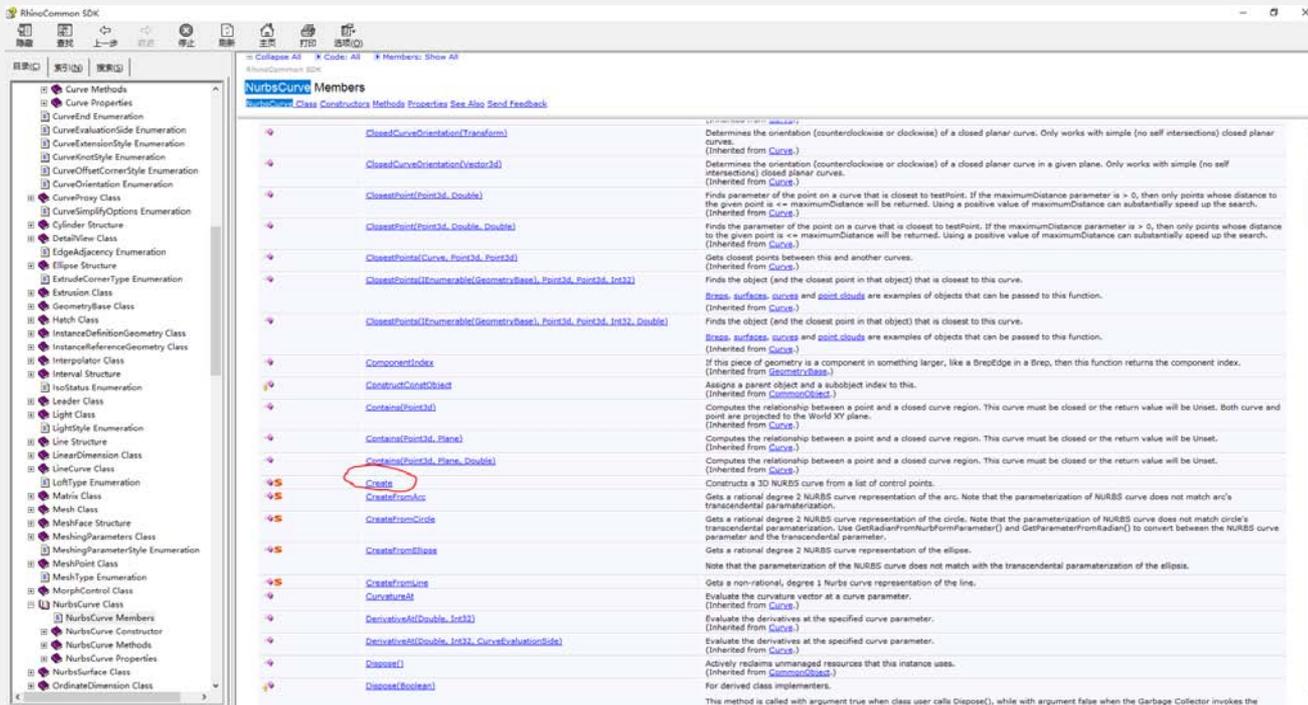
public class Component_NurbsCurve : GH_Component
Name: CurveComponents.Component_NurbsCurve
Assembly: CurveComponents, Version=1.0.0.0

2.6 函数调用关系

注意其函数实现中有这样一句话：

```
NurbsCurve data = NurbsCurve.Create(false, degree, list);
```

这个 `NurbsCurve.Create()` 就是Rhino自身的函数，gh只是调用了它，并加了一些接口与类型自动转换的函数，并打包成了一个电池，其中，`false`就是是否闭合，`degree`就是曲线的阶数，`list`就是控制点的List而这个函数的实现在RhinoCommon中可以查到：



2.6 函数调用关系

NurbsCurve.Create(){}

包括一些例子

NurbsCurve.Create Method

[NurbsCurve Class Example](#) [See Also](#) [Send Feedback](#)

Constructs a 3D NURBS curve from a list of control points.

Namespace: [Rhino.Geometry](#)

Assembly: RhinoCommon (in RhinoCommon.dll) Version: 5.1.30000.5 (5.0.20693.0)

Syntax

C#

```
public static NurbsCurve Create(  
    bool periodic,  
    int degree,  
    IEnumerable<Point3d> points  
)
```

Visual Basic

```
Public Shared Function Create ( _  
    periodic As Boolean, _  
    degree As Integer, _  
    points As IEnumerable(Of Point3d) _  
) As NurbsCurve
```

Parameters

periodic

Type: [System.Boolean](#)

If true, create a periodic uniform curve. If false, create a clamped uniform curve.

degree

Type: [System.Int32](#)

(>=1) degree=order-1.

points

Type: [System.Collections.Generic.IEnumerable\(Point3d\)](#)

control vertex locations.

Return Value

new NURBS curve on success null on error.

Examples

Examples

VB.NET

Partial Class Examples

```
Public Shared Function AddNurbsCurve(ByVal doc As Rhino.RhinoDoc) As Rhino.Commands.Result  
    Dim points As New Rhino.Collections.Point3dList(5)  
    points.Add(0, 0, 0)  
    points.Add(0, 2, 0)  
    points.Add(2, 3, 0)  
    points.Add(4, 2, 0)  
    points.Add(4, 0, 0)  
    Dim nc As Rhino.Geometry.NurbsCurve = Rhino.Geometry.NurbsCurve.Create(False, 3, points)  
    Dim rc As Rhino.Commands.Result = Rhino.Commands.Result.Failure  
    If nc IsNot Nothing AndAlso nc.IsValid Then  
        If doc.Objects.AddCurve(nc) <> Guid.Empty Then  
            doc.Views.Redraw()  
            rc = Rhino.Commands.Result.Success  
        End If  
    End If  
    Return rc  
End Function  
End Class
```

C#

using System;

partial class Examples

```
{  
    public static Rhino.Commands.Result AddNurbsCurve(Rhino.RhinoDoc doc)  
    {  
        Rhino.Collections.Point3dList points = new Rhino.Collections.Point3dList(5);  
        points.Add(0, 0, 0);  
        points.Add(0, 2, 0);  
        points.Add(2, 3, 0);  
        points.Add(4, 2, 0);  
        points.Add(4, 0, 0);  
        Rhino.Geometry.NurbsCurve nc = Rhino.Geometry.NurbsCurve.Create(false, 3, points);  
        Rhino.Commands.Result rc = Rhino.Commands.Result.Failure;  
        if (nc != null && nc.IsValid)  
        {  
            if (doc.Objects.AddCurve(nc) != Guid.Empty)  
            {  
                doc.Views.Redraw();  
                rc = Rhino.Commands.Result.Success;  
            }  
        }  
        return rc;  
    }  
}
```

Python

```
import Rhino  
import scriptcontext  
import System.Guid
```

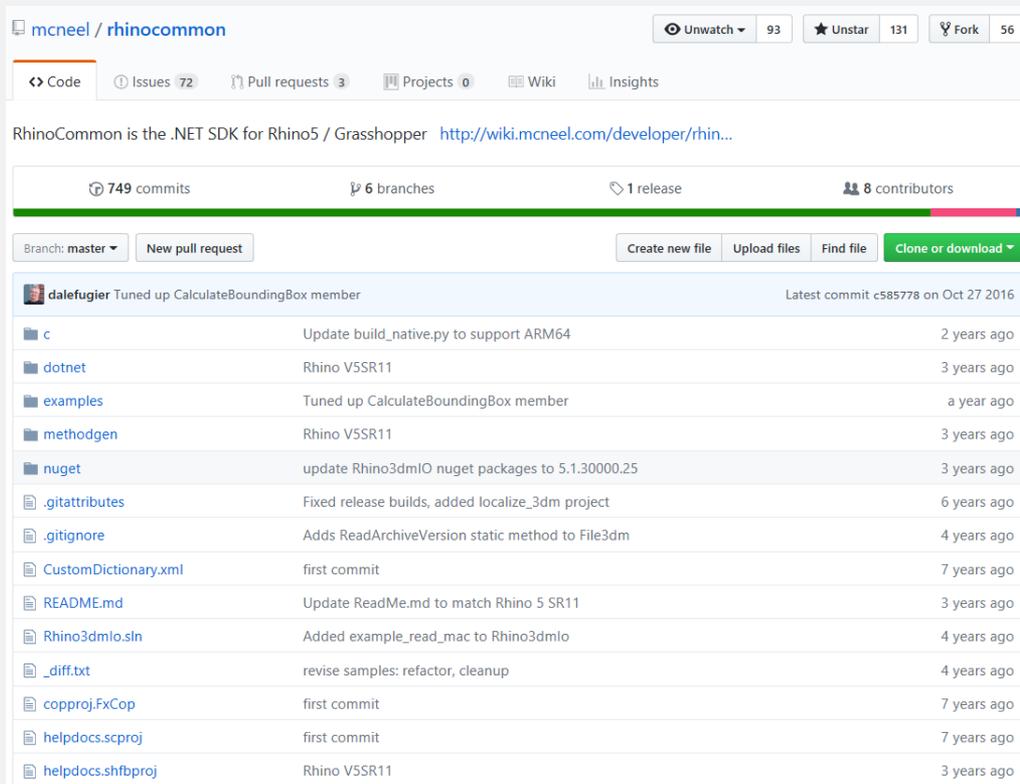
2.6 函数调用关系

NurbsCurve.Create()的实现可以在Github上查到

<https://github.com/mcneel/rhinocommon>

官网链接

<http://developer.rhino3d.com/guides/rhinocommon/what-is-rhinocommon/>



mcneel / rhinocommon

Unwatch 93 Unstar 131 Fork 56

Code Issues 72 Pull requests 3 Projects 0 Wiki Insights

RhinoCommon is the .NET SDK for Rhino5 / Grasshopper <http://wiki.mcneel.com/developer/rhin...>

749 commits 6 branches 1 release 8 contributors

Branch: master New pull request Create new file Upload files Find file Clone or download

dalefugier Tuned up CalculateBoundingBox member Latest commit c585778 on Oct 27 2016

c	Update build_native.py to support ARM64	2 years ago
dotnet	Rhino V5SR11	3 years ago
examples	Tuned up CalculateBoundingBox member	a year ago
methodgen	Rhino V5SR11	3 years ago
nuget	update Rhino3dmIO nuget packages to 5.1.30000.25	3 years ago
.gitattributes	Fixed release builds, added localize_3dm project	6 years ago
.gitignore	Adds ReadArchiveVersion static method to File3dm	4 years ago
CustomDictionary.xml	first commit	7 years ago
README.md	Update ReadMe.md to match Rhino 5 SR11	3 years ago
Rhino3dmIo.sln	Added example_read_mac to Rhino3dmIo	4 years ago
_diff.txt	revise samples: refactor, cleanup	4 years ago
copproj.FxCop	first commit	7 years ago
helpdocs.scproj	first commit	7 years ago
helpdocs.shfbproj	Rhino V5SR11	3 years ago

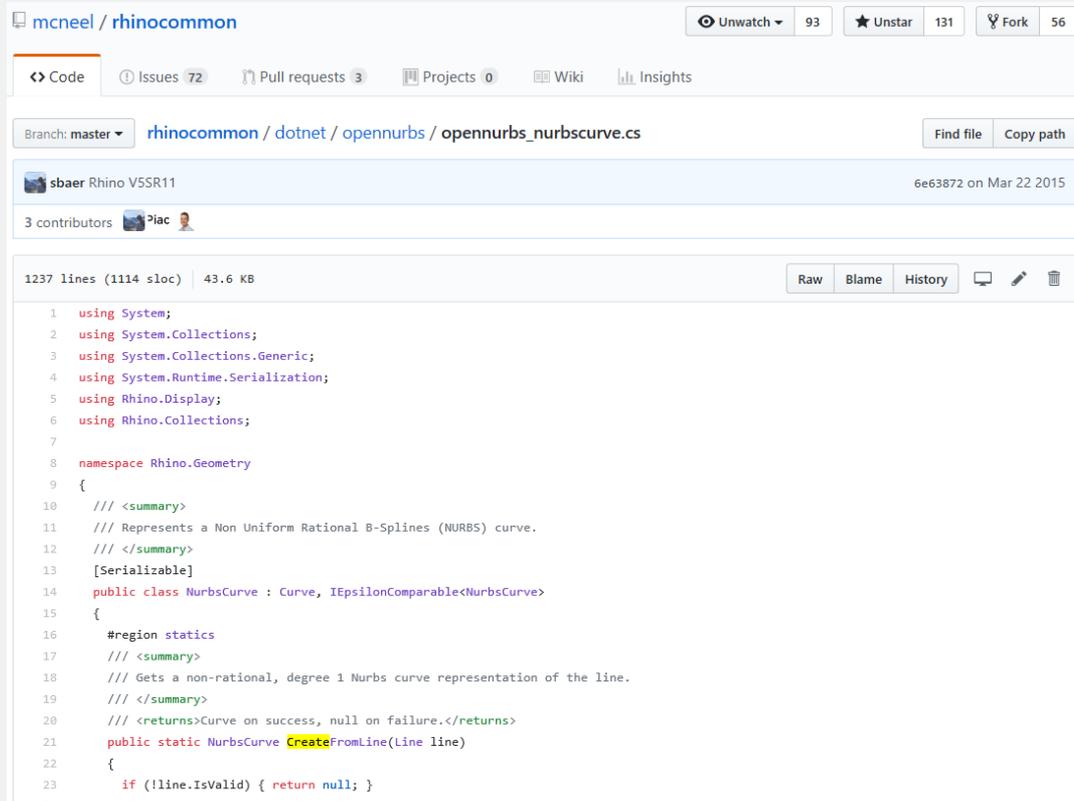
2.6 函数调用关系

NurbsCurve.Create()的实现可以在Github上查到

<https://github.com/mcneel/rhinocommon> 的第137行

官网链接

<http://developer.rhino3d.com/guides/rhinocommon/what-is-rhinocommon/>



The screenshot shows the GitHub repository page for mcneel/rhinocommon. The repository is on the master branch, and the file path is rhinocommon / dotnet / opennurbs / opennurbs_nurbscurve.cs. The commit is 6e63872, dated Mar 22 2015, by sbaer. The code is for the NurbsCurve class, specifically the CreateFromLine method. The code is as follows:

```
1 using System;
2 using System.Collections;
3 using System.Collections.Generic;
4 using System.Runtime.Serialization;
5 using Rhino.Display;
6 using Rhino.Collections;
7
8 namespace Rhino.Geometry
9 {
10     /// <summary>
11     /// Represents a Non Uniform Rational B-Splines (NURBS) curve.
12     /// </summary>
13     [Serializable]
14     public class NurbsCurve : Curve, IEpsilonComparable<NurbsCurve>
15     {
16         #region statics
17         /// <summary>
18         /// Gets a non-rational, degree 1 Nurbs curve representation of the line.
19         /// </summary>
20         /// <returns>Curve on success, null on failure.</returns>
21         public static NurbsCurve CreateFromLine(Line line)
22         {
23             if (!line.IsValid) { return null; }
24         }
25     }
26 }
```

2.6 函数调用关系

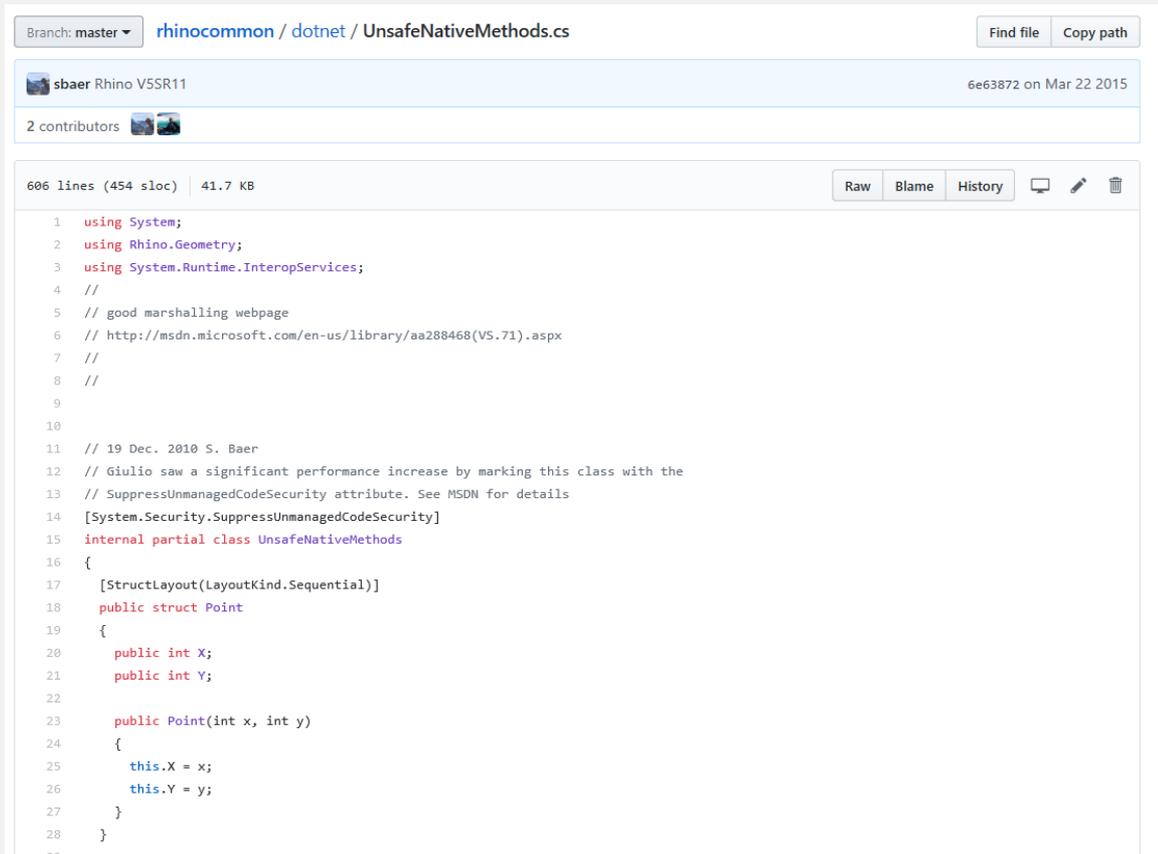
NurbsCurve.Create()

```
137     public static NurbsCurve Create(bool periodic, int degree, System.Collections.Generic.IEnumerable<Point3d> points)
138     {
139         if (degree < 1)
140             return null;
141
142         const int dimension = 3;
143         const double knot_delta = 1.0;
144         int count;
145         int order = degree + 1;
146         Point3d[] point_array = Point3dList.GetConstPointArray(points, out count);
147         if (null == point_array || count < 2)
148             return null;
149
150         NurbsCurve nc = new NurbsCurve();
151         IntPtr ptr_curve = nc.NonConstPointer();
152
153         bool rc = periodic ? UnsafeNativeMethods.ON_NurbsCurve_CreatePeriodicUniformNurbs(ptr_curve, dimension, order, count, point_array, kn
154                               UnsafeNativeMethods.ON_NurbsCurve_CreateClampedUniformNurbs(ptr_curve, dimension, order, count, point_array, kno
155
156         if (false == rc)
157         {
158             nc.Dispose();
159             return null;
160         }
161         return nc;
162     }
```

2.6 函数调用关系

Create()函数内调用了一个UnsafeNativeMethods.cs的类，而在这个类内部，引入了一些外部的C/C++函数 ON_NurbsCurve_CreateClampedUniformNurbs

<https://github.com/mcneel/rhinocommon/blob/master/dotnet/UnsafeNativeMethods.cs>

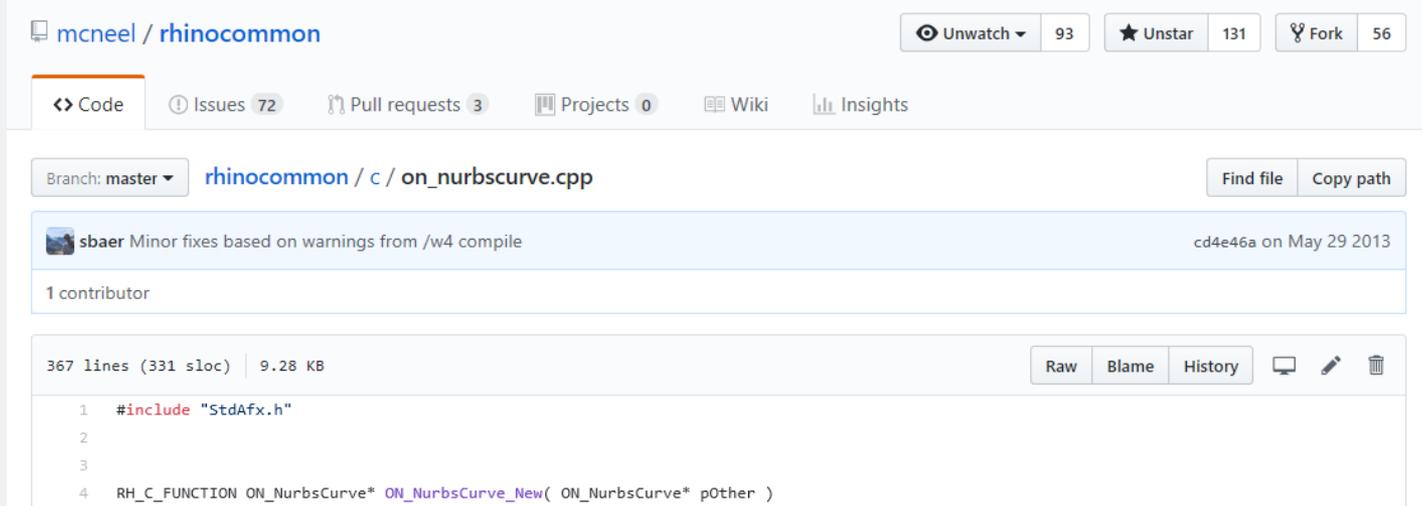


The screenshot shows a GitHub file viewer for the file `UnsafeNativeMethods.cs` in the `rhinocommon / dotnet` directory. The file is 606 lines long (454 sloc) and 41.7 KB. It is a C# file that defines a partial class `UnsafeNativeMethods` with a `Point` struct. The code includes several using statements and a comment about performance improvements.

```
1 using System;
2 using Rhino.Geometry;
3 using System.Runtime.InteropServices;
4 //
5 // good marshalling webpage
6 // http://msdn.microsoft.com/en-us/library/aa288468(VS.71).aspx
7 //
8 //
9
10
11 // 19 Dec. 2010 S. Baer
12 // Giulio saw a significant performance increase by marking this class with the
13 // SuppressUnmanagedCodeSecurity attribute. See MSDN for details
14 [System.Security.SuppressUnmanagedCodeSecurity]
15 internal partial class UnsafeNativeMethods
16 {
17     [StructLayout(LayoutKind.Sequential)]
18     public struct Point
19     {
20         public int X;
21         public int Y;
22
23         public Point(int x, int y)
24         {
25             this.X = x;
26             this.Y = y;
27         }
28     }
29 }
```

2.6 函数调用关系

https://github.com/mcneel/rhinocommon/blob/master/c/on_nurbscurve.cpp



mcneel / rhinocommon

Unwatch 93 Unstar 131 Fork 56

Code Issues 72 Pull requests 3 Projects 0 Wiki Insights

Branch: master rhinocommon / c / on_nurbscurve.cpp Find file Copy path

sbaer Minor fixes based on warnings from /w4 compile cd4e46a on May 29 2013

1 contributor

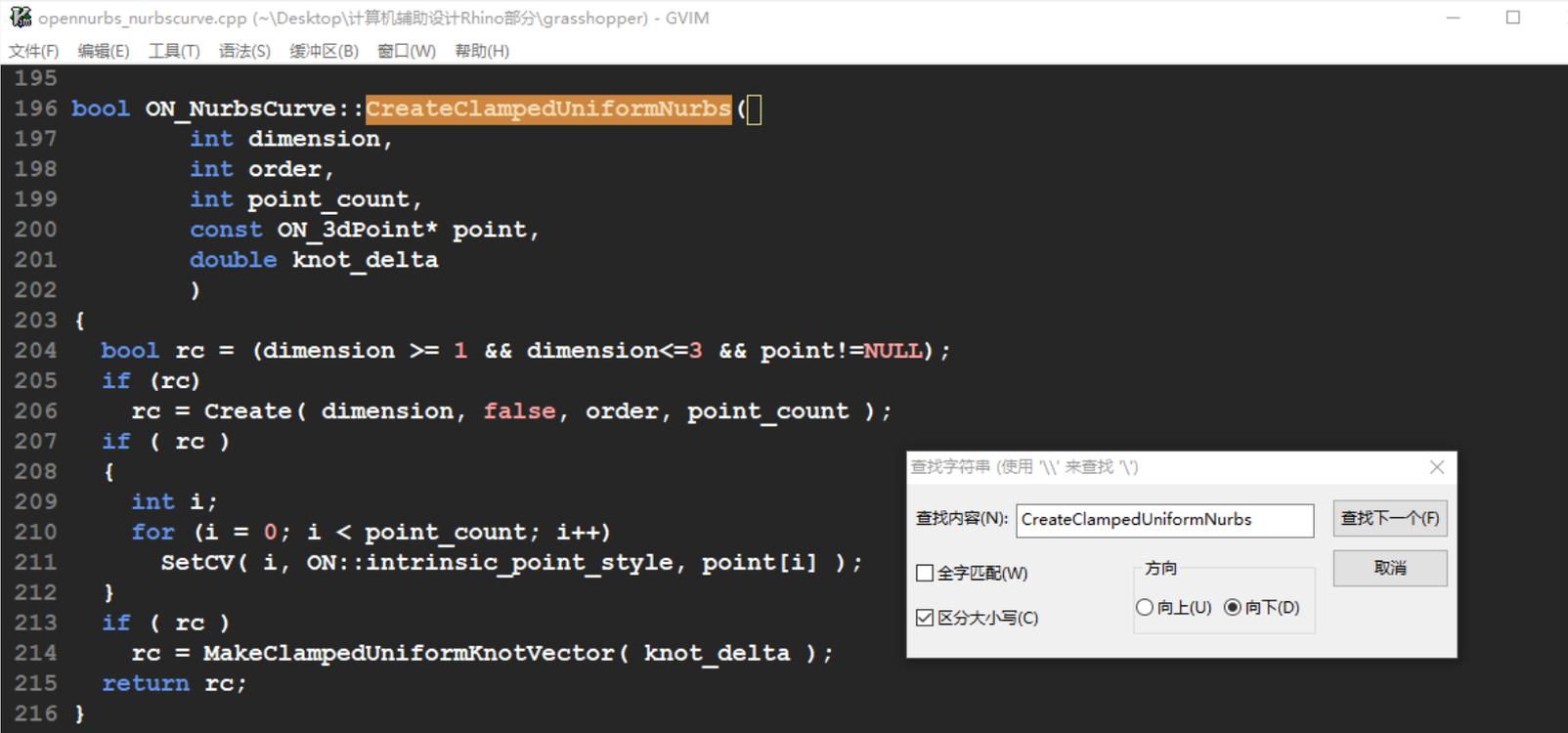
367 lines (331 sloc) | 9.28 KB Raw Blame History

```
1 #include "StdAfx.h"
2
3
4 RH_C_FUNCTION ON_NurbsCurve* ON_NurbsCurve_New( ON_NurbsCurve* pOther )
```

```
25 RH_C_FUNCTION bool ON_NurbsCurve_CreateClampedUniformNurbs(ON_NurbsCurve* crv, int dim, int order, int count, /*ARRAY*/const ON_3dPoint* pts
26 {
27     if( NULL==crv || NULL == pts ) CreateClampedUniformNurbs
28         return false;
29     const ON_3dPoint* _pts = (const ON_3dPoint*)pts;
30     return crv->CreateClampedUniformNurbs(dim, order, count, _pts, knot_delta);
31 }
32
33 RH_C_FUNCTION bool ON_NurbsCurve_CreatePeriodicUniformNurbs(ON_NurbsCurve* crv, int dim, int order, int count, /*ARRAY*/const ON_3dPoint* p
34 {
35     if( NULL==crv || NULL == pts )
36         return false;
37     return crv->CreatePeriodicUniformNurbs(dim, order, count, pts, knot_delta);
38 }
```


2.6 函数调用关系

crv->CreateClampedUniformNurbs



The image shows a code editor window with a dark background. The code is in C++ and defines a function `CreateClampedUniformNurbs` for the `ON_NurbsCurve` class. The function signature is `bool ON_NurbsCurve::CreateClampedUniformNurbs(int dimension, int order, int point_count, const ON_3dPoint* point, double knot_delta)`. The function body includes a check for valid dimensions, a call to `Create`, a loop to set point styles, and a call to `MakeClampedUniformKnotVector`. A search dialog box is overlaid on the code, showing the search content `CreateClampedUniformNurbs` and search options.

```
195
196 bool ON_NurbsCurve::CreateClampedUniformNurbs(
197     int dimension,
198     int order,
199     int point_count,
200     const ON_3dPoint* point,
201     double knot_delta
202 )
203 {
204     bool rc = (dimension >= 1 && dimension<=3 && point!=NULL);
205     if (rc)
206         rc = Create( dimension, false, order, point_count );
207     if ( rc )
208     {
209         int i;
210         for ( i = 0; i < point_count; i++)
211             SetCV( i, ON::intrinsic_point_style, point[i] );
212     }
213     if ( rc )
214         rc = MakeClampedUniformKnotVector( knot_delta );
215     return rc;
216 }
```

查找字符串 (使用 '\\' 来查找 '\')

查找内容(N): CreateClampedUniformNurbs 查找下一个(F)

全字匹配(W) 方向 向上(U) 向下(D)

区分大小写(C) 取消

2.6 函数调用关系

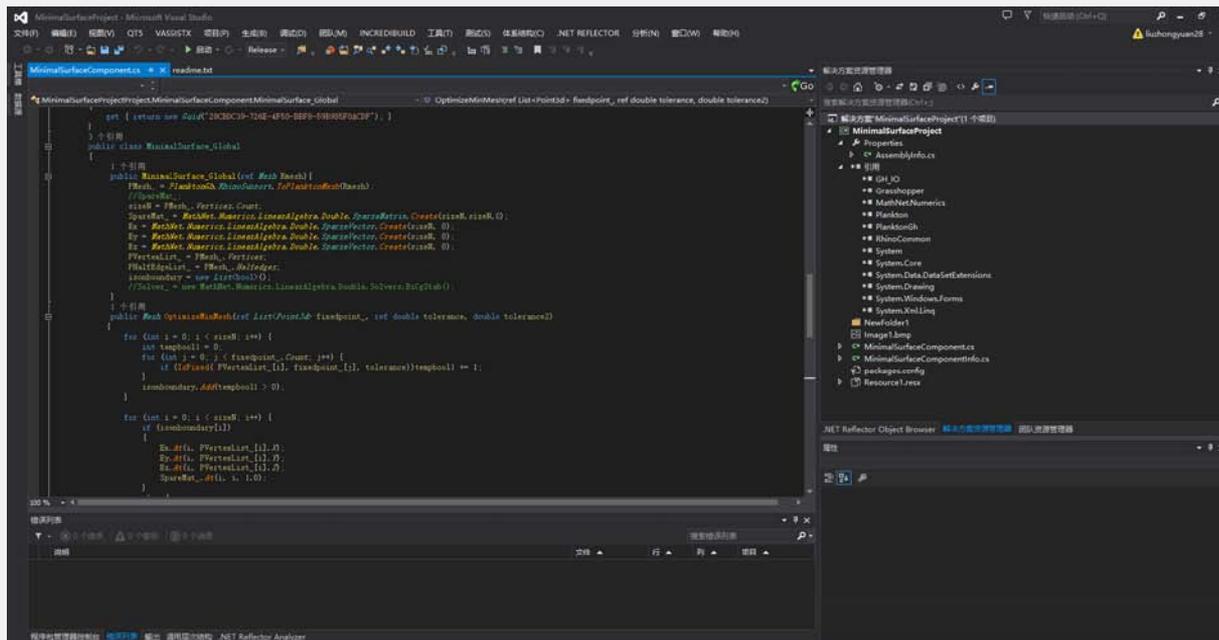
crv->Create

```
242 bool ON_NurbsCurve::Create(  
243     int dim,           // dimension (>= 1)  
244     ON_BOOL32 is_rat, // true to make a rational NURBS  
245     int order,        // order (>= 2)  
246     int cv_count      // cv count (>= order)  
247 )  
248 {  
249     DestroyCurveTree();  
250     if ( dim < 1 )  
251         return false;  
252     if ( order < 2 )  
253         return false;  
254     if ( cv_count < order )  
255         return false;  
256     m_dim = dim;  
257     m_is_rat = (is_rat) ? true : false;  
258     m_order = order;  
259     m_cv_count = cv_count;  
260     m_cv_stride = (m_is_rat) ? m_dim+1 : m_dim;  
261     bool rc = ReserveKnotCapacity( KnotCount() );  
262     if ( !ReserveCVCapacity( CVCCount()*m_cv_stride ) )  
263         rc = false;  
264     return rc;  
265 }  
266
```



2.7 C#电池编程与引用

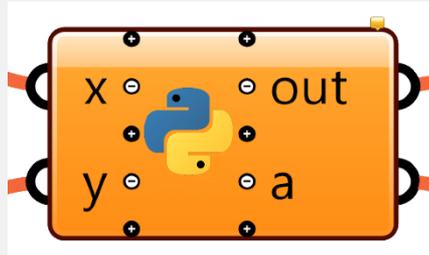
- 用vs打开MinimalSurfaceComponentExample
- 可以看见在gh内使用Plankon与mathnet矩阵库实现的极小曲面案例
- 注意引用的库与自定义的gha文件的存放位置，存放位置查看gh菜单说明.jpg



2.8 C#电池Pinvoke调用NativeC++

- 1.查看工程案例Pinvoketest2 练习c#与C++间传递结构体指针。
- 2.查看工程MiniMeshdll参考对Mesh的转换与传递。
C++内生开辟的内存空间可在dll中做一个释放内存的函数接口，gh电池运行完之前调用进行释放，可避免内存泄漏。
- 提示:通过Pinvoke调用c++dll，可以把任意外部C++库在rhino内使用，亲测可用:Eigen、MKL、OpenMesh、Libigl、CGAL、Qt等，注意调用时要配好特点依赖的dll到Rhino环境变量中，让Rhino能找到，懒的话可以粗暴的扔进system32内。。。

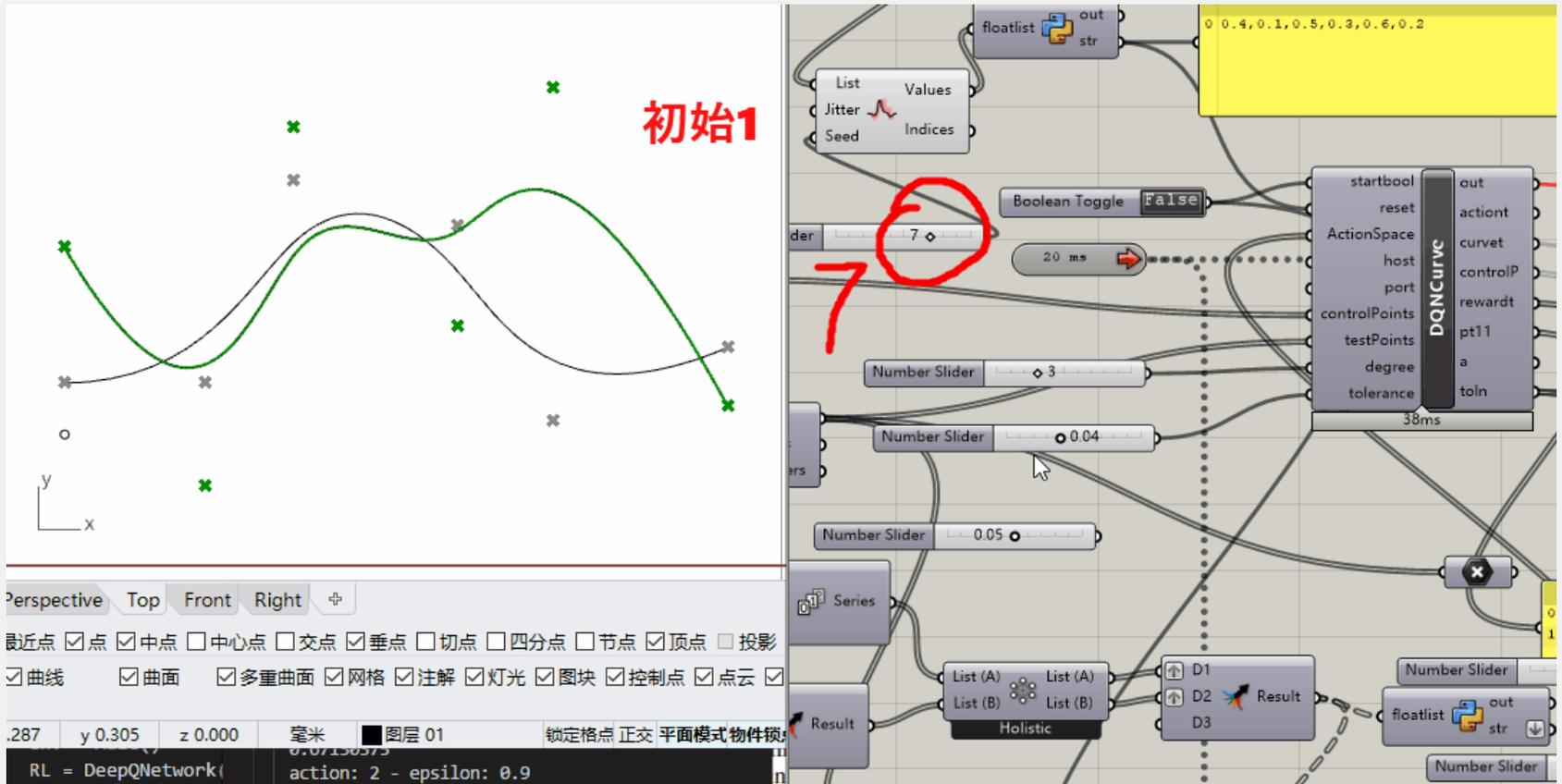
2.9.1 IronPython2.7.8.0



- 1:IronPython2.7.8.0 (gh自带)
- 特点: rhino官方维护, 支持rhino、grasshopper object与gh数据结构。不支持外部python库, 如tensorflow、numpy等。帮助:
- <https://developer.rhino3d.com/guides/rhinopython/>
- <https://developer.rhino3d.com/guides/rhinopython/your-first-python-script-in-grasshopper/>

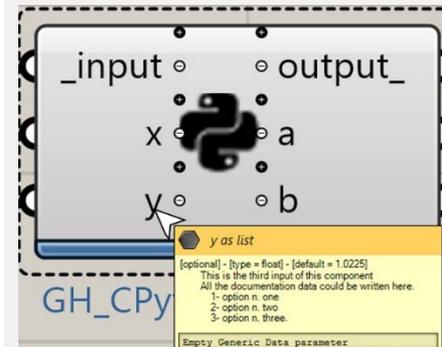
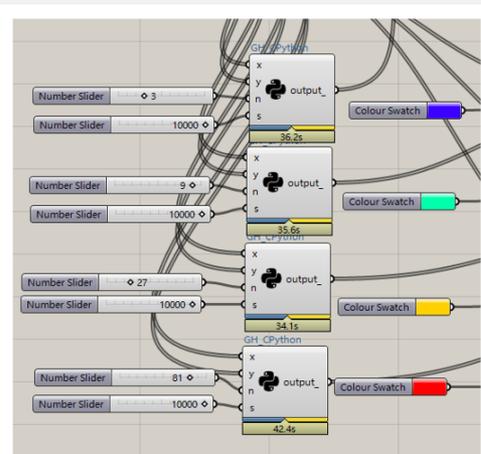
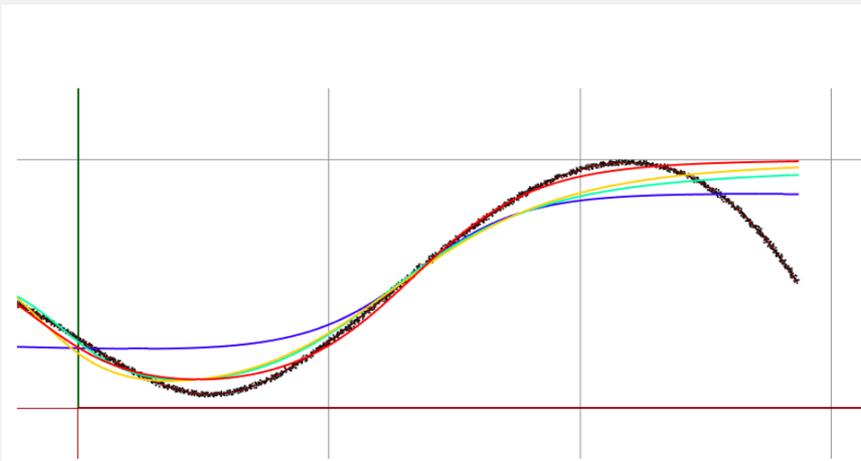
2.9.1 IronPython2.7.8.0

- 虽然IronPython不支持外部库，但其可用大部分rhino、gh函数，并且支持gh的数据结构，且响应迅速，可以使用python通信机制与外部python、C++程序交互（甚至利用TCP通信与其他电脑传输数据）。
- 下图为IronPython与另一台服务器交互数据，在服务器上进行强化学习训练拟合曲线



2.9.2 Gh_Cpython

- 下载地址：https://github.com/MahmoudAbdelRahman/GH_CPython
- 特点：本质上使用的是外部Python解释器，电池本身只是一个txt编辑器+数据传输接口，会把内部代码存成.py文件并调用配置好的外部python.exe运行，并把接口传入的数据送给外部python.exe，并且接收回返回数据。所以很难使用rhino对象，需要把rhino对象转成pythonlist（[]）等传入处理，并且相应速度较慢。但可以使用任意外部python库，如tensorflow、numpy，与任意pytho版本，2.7、2.8、3.6、3.7等。详细看github.com源代码。实际上速度慢，需要跑完全程，不建议使用。
- 下图直接在gh中使用Gh_Cpython调用tensorflow与numpy训练神经网络拟合点云。



2.10 cmd.exe

- 可以使用C#电池直接在rhino中调用cmd.exe，运行任何exe！代码如下：

```
private void RunScript(string exefullpath, string argv1, string argv2, ref object A)
{
    string command, output;
    command = exefullpath + " " + argv1 + " " + argv2;
    RunCmd.run(command, out output);
    A = output;
}

// <Custom additional code>
public class RunCmd
{
    public RunCmd()
    {
    }

    private static string CmdPath = @"C:\Windows\System32\cmd.exe";

    /// <summary>
    /// 执行cmd命令
    /// 多命令请使用批处理命令连接符：
    /// <![CDATA[
    /// & 同时执行两个命令
    /// ]:将上一个命令的输出,作为下一个命令的输入
    /// &&: 当&&前的命令成功时,才执行&&后的命令
    /// ||: 当||前的命令失败时,才执行||后的命令]>
    /// 其他请百度
    /// </summary>
    /// <param name="cmd"></param>
    /// <param name="output"></param>
    public static void run(string cmd, out string output)
    {
        cmd = cmd.Trim().TrimEnd('&') + " &exit"; //说明：不管命令是否成功均执行exit命令，否则当调用ReadToEnd()方法时，会处于假死状态
        using (Process p = new Process())
        {
            p.StartInfo.FileName = CmdPath;
            p.StartInfo.UseShellExecute = false; // 是否使用操作系统shell启动
            p.StartInfo.RedirectStandardInput = true; //接受来自调用程序的输入信息
            p.StartInfo.RedirectStandardOutput = true; //由调用程序获取输出信息
            p.StartInfo.RedirectStandardError = true; //重定向标准错误输出
            p.StartInfo.CreateNoWindow = true; //不显示程序窗口
            p.Start(); //启动程序

            //向cmd窗口写入命令
            p.StandardInput.WriteLine(cmd);
            p.StandardInput.AutoFlush = true;
            p.StandardInput.WriteLine("exit");

            //向标准输入写入要执行的命令。这里使用&是批处理命令的符号，表示前面一个命令不管是否执行成功都执行后面(exit)命令，如果不执行exit命令，后面调用ReadToEnd()方法会假死
            //同类的符号还有&&和||前者表示必须前一个命令执行成功才会执行后面的命令，后者表示必须前一个命令执行失败才会执行后面的命令

            //获取cmd窗口的输出信息

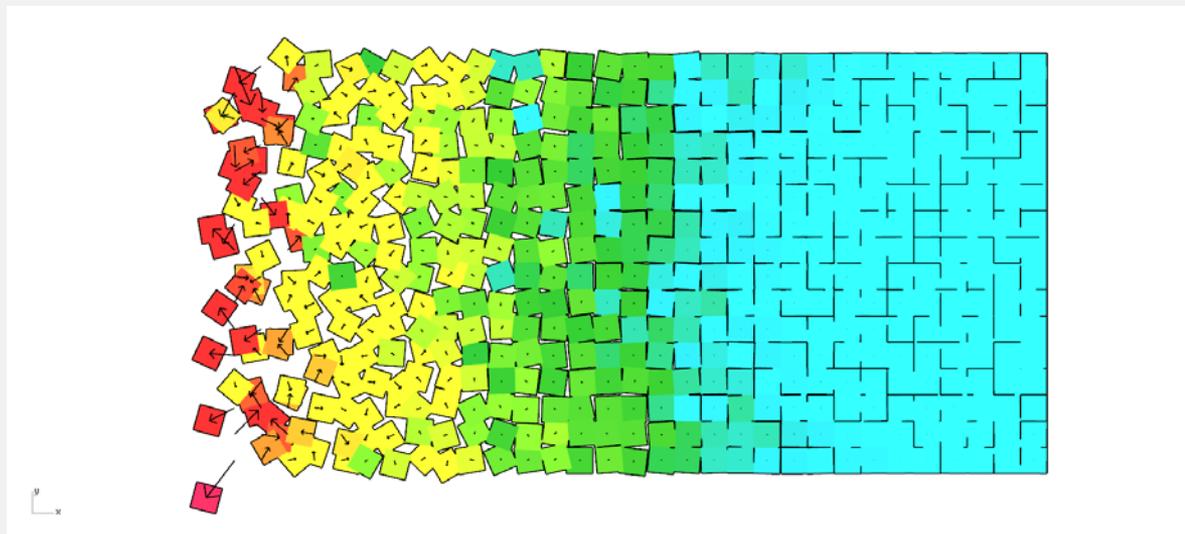
            output = p.StandardOutput.ReadToEnd();

            //p.WaitForExit();//等待程序执行完退出进程
            p.Close();
        }
    }
}

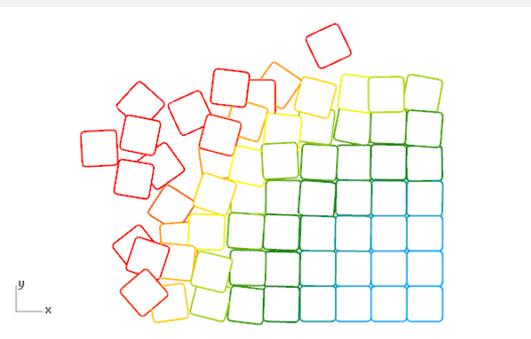
// </Custom additional code>
```

3.1 查看gh关联模型案例、动画案例等练习

- 打开2.gh文件
- 尝试理解电池逻辑并复制出一样的电池实现，之后看set综合.gh



set综合.gh



练习：基于C++的Grasshopper插件制作

- 1. 要求：
- 使用vs的gh模板制作自己的电池插件，并调用C++dll，