C++编写 Grasshopper 插件

QQ 164848615 杭州萧山 2015.07.01

Grasshopper 的插件系统是基于.Net Framework 的，用纯 C++编写插件比较繁琐，这儿考虑接口部分使用 C++\CLI 编写，核心算法还是使用 C++，毕竟有关图形、几何方面的算法或建模库，大多是 C++编写的，比如 CGAL，boost.geometry，OpenCASCADE…… 事实上，Rhino 本身核心也是 C++编写的，C++是工业基础编程语言。

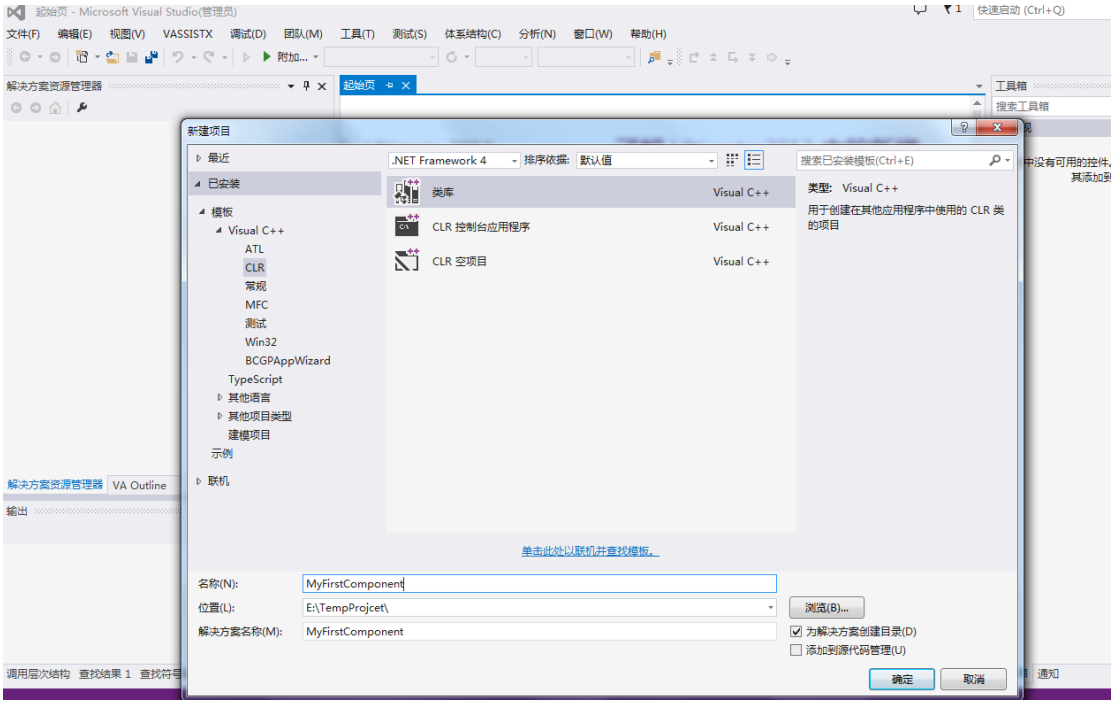
C++\CLI 是微软针对.Net Framework 为 C++开发者提供的非 ISO 扩展，截至到目前，ISO C++还没有完善的 ABI 解决方案，微软自己动手，解决了边界安全问题，方便 C++用户使用.Net 框架。C++\CLI 也经历了多个版本发展，目前 Visual Studio 2013 带的 C++\CLI 已经较为完善了，本文将以这个版本作为演示例子。可以预见的是，C++\CLI 也是一个过渡版本，Visual Studio 2015 发展到 C++\CX，保证边界安全的同时，是语法更接近 ISO C++，更加方便 ISO C++开发人员。在即将投票的 ISO C++17 标准中，ABI 已经作为一个关键项多次讨论了，也就是说，C++一个平台编译，多个平台运行的情况很可能就要成真了。

C++编译工具直接将源代码编译成机器码，所以根据 CPU 不同类型，会生成不同的代码，通常会有 x86 和 x64 两种，运行于常规的 32 位或 64 位电脑。所以，用 C++编写 GH 插件的时候，一定要分清，这点与 GH 自带的 C#开发例子不同。

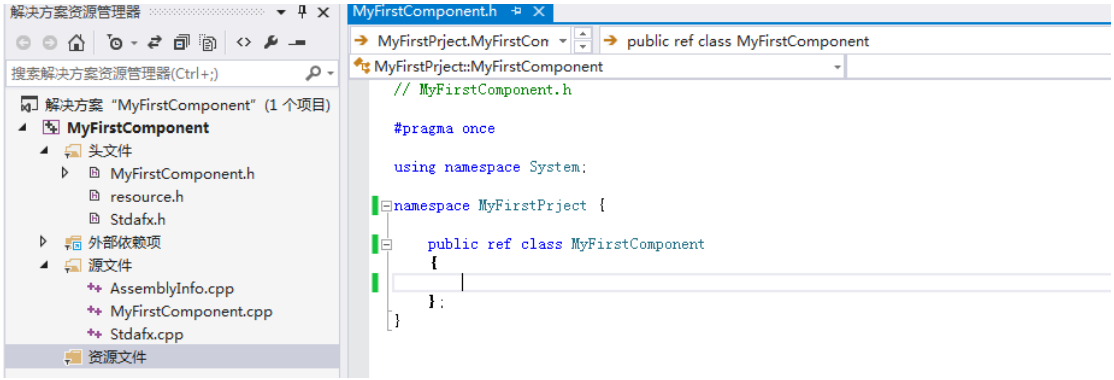
还有一点要提前说明，微软的 C++\CLI 混合编程时，不允许静态链接，发布时，需要目标计算机装了对应版本的 VC++运行库。

下面我演示一下 Grasshopper SDK （下面简称 SDK）自带的例子，但语言改成 C++\CLI。

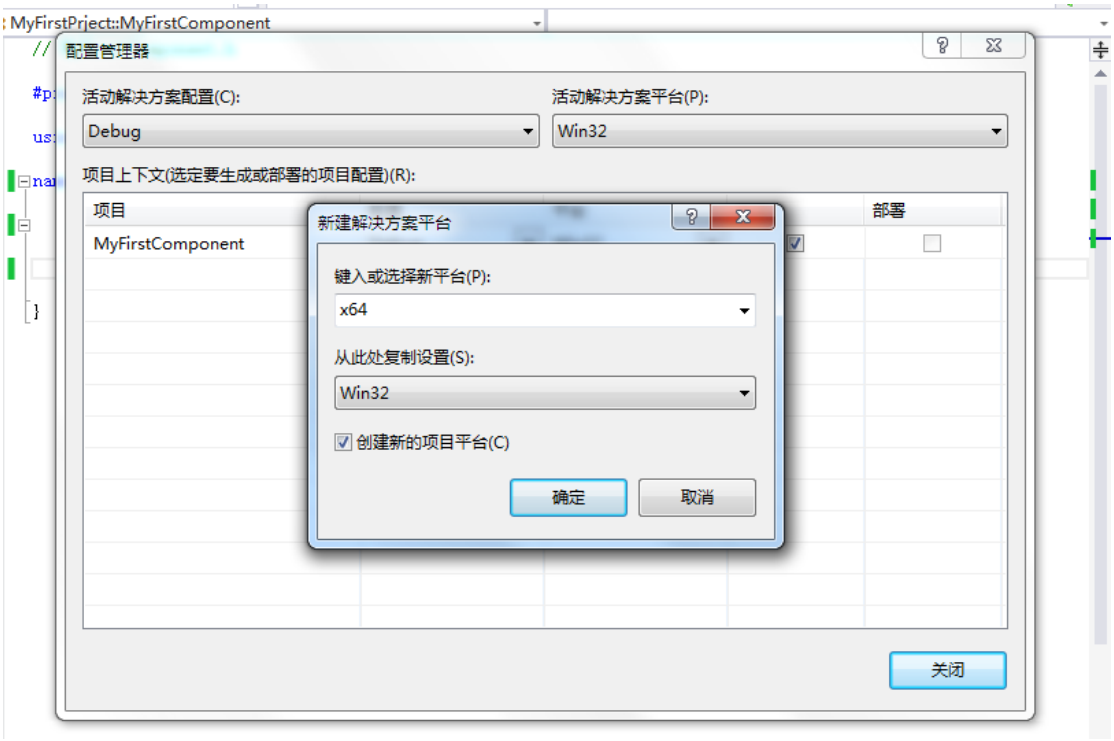
1. 新建一个 CLR 类库，名称与 SDK 一样，叫 MyFirstComponent



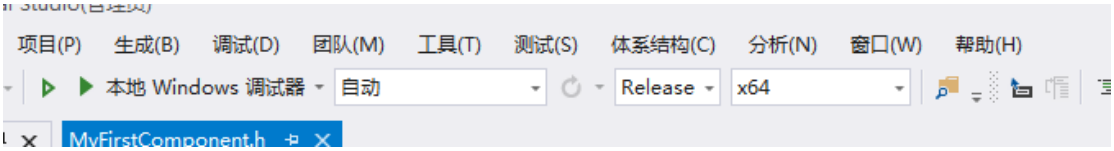
改一下类名和命名空间名



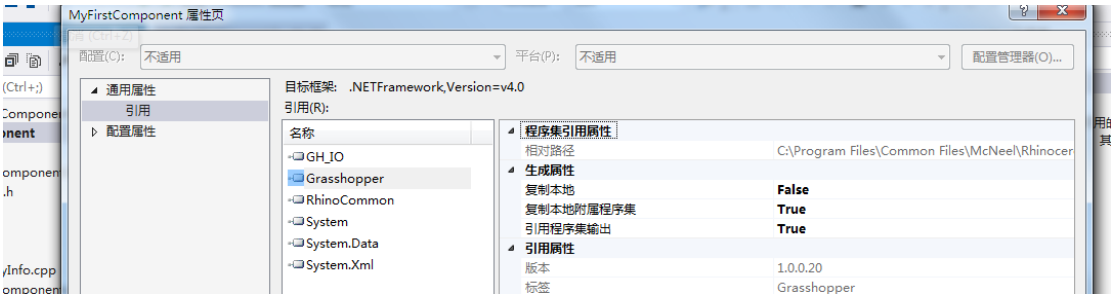
2. 由于演示电脑是 64 位的，Rhino 也装的是 64 位，所以这儿我先将平台配置到 x64



这儿也不调试了，直接使用 Release：



3. 添加引用 RhinoCommon.dll，Grasshopper.dll，GH_IO.dll。Copy Local（复制本地）均为 false



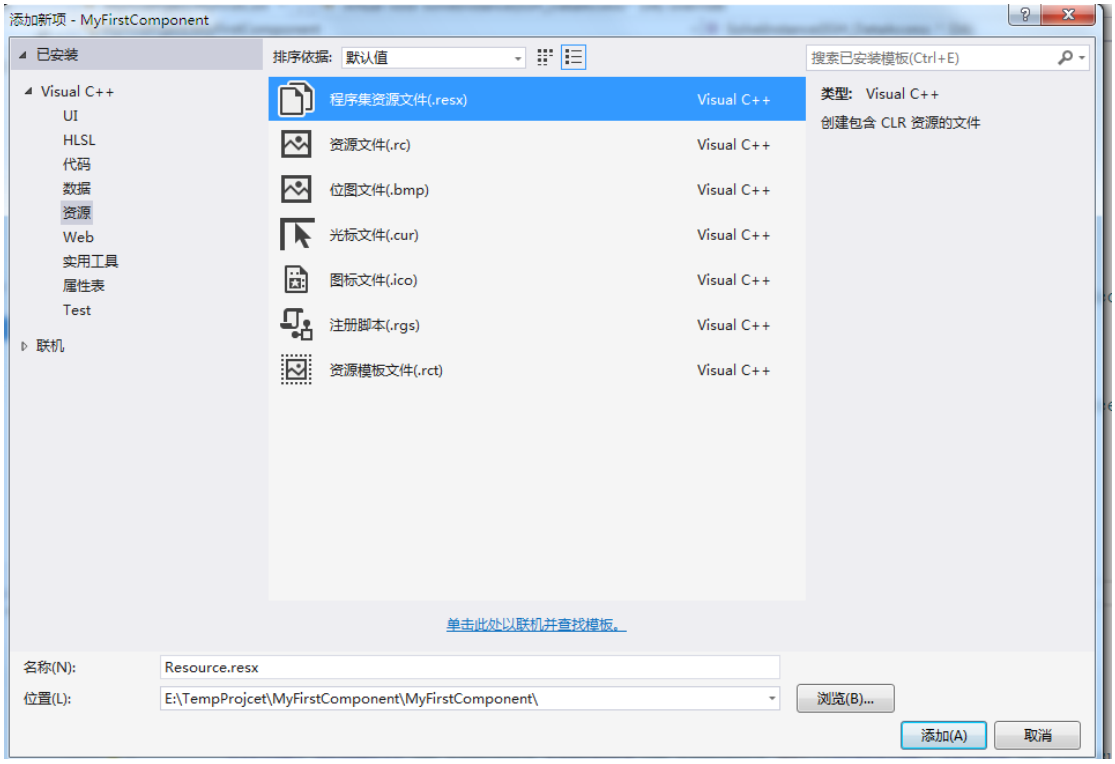
4. 加入引用代码，修改基类，构造函数，利用 VS 工具创建一个 GUID：

```
MyFirstPrject::MyFirstComponent
// MyFirstComponent.h

#pragma once
#include <msclr\marshal.h>
#include <msclr\marshal_cppstd.h>
using namespace System;
using namespace System::Drawing;
using namespace System::Windows::Forms;
using namespace System::Collections;
using namespace System::Collections::Generic;
using namespace Rhino;
using namespace Rhino::Geometry;
using namespace Rhino::Input;
using namespace Rhino::DocObjects;
using namespace Grasshopper::Kernel;
using namespace Grasshopper::Kernel::Types;
using namespace System;

namespace MyFirstPrject {
    public ref class MyFirstComponent :public GH_Component
    {
    public:
        MyFirstComponent()
            :GH_Component(L"MyFirst", L"MFC", L"My first component", L"Extra", L"Simple")
        {}
        virtual property Guid ComponentGuid {
            Guid get() override{
                return Guid("{80CF29AD-A499-48DD-9C3D-8917CD3A34C0}"); };
        }
```

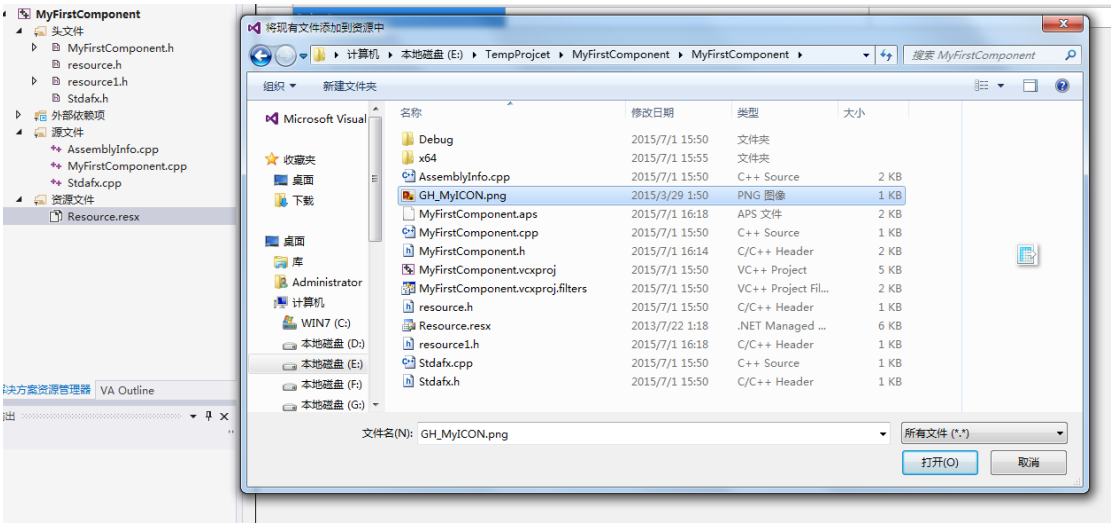
5. 继续添加输入、输出、计算过程以及插件在 GH 界面上的显示方式：

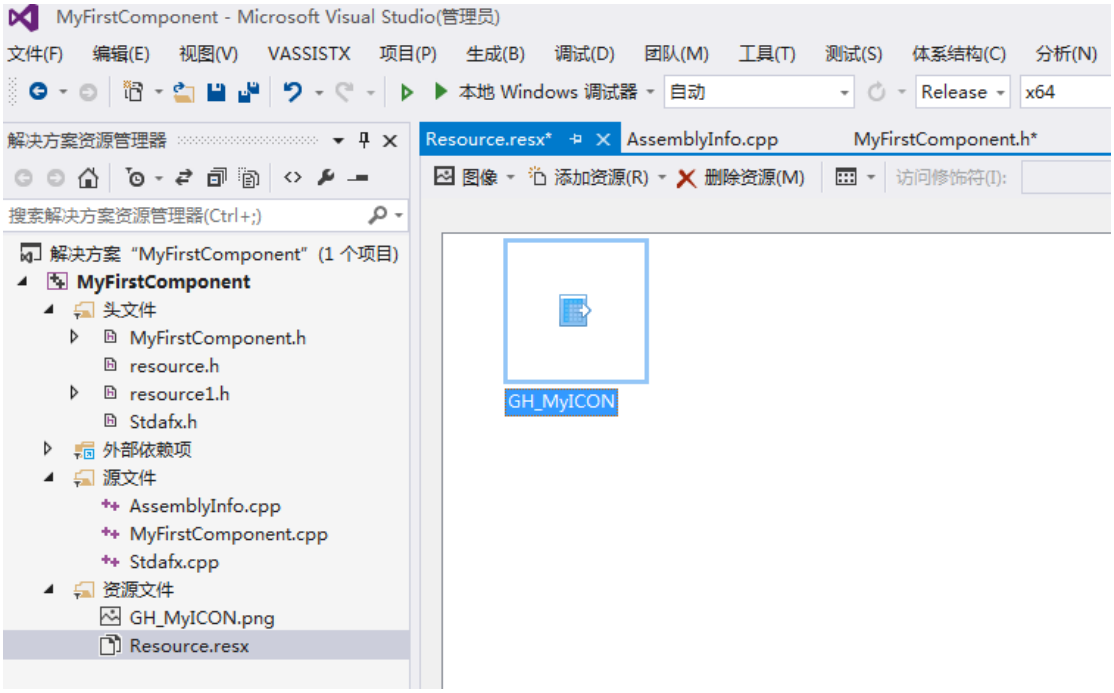
```
namespace MyFirstPrject {
    public ref class MyFirstComponent :public GH_Component
    {
    public:
        MyFirstComponent() { ... }
        virtual property Guid ComponentGuid { ... }
        virtual void RegisterInputParams(GH_InputParamManager^ pManager) override
        {
            pManager->AddTextParameter(L"String", L"S", L"String to reverse", GH_ParamAccess::item);
        }
        virtual void RegisterOutputParams(GH_OutputParamManager^ pManager) override
        {
            pManager->AddTextParameter(L"Reverse", L"R", L"Reversed string", GH_ParamAccess::item);
        }
        virtual void SolveInstance(IGH_DataAccess^ DA) override
        {
            String ^data = nullptr;
            if (!DA->GetData(0, data)) return;
            if (data == nullptr) return;
            if (data->Length==0) return;

            auto chars = data->ToCharArray();
            System::Array::Reverse(chars);
            DA->SetData(0, gcnew String(chars));
        }
        virtual property GH_Exposure Exposure{GH_Exposure get()override{ return GH_Exposure::primary; }}
    };
}
```

6. 删除自动生成的资源，添加“程序集资源文件”，



准备一个 24x24 的 PNG 图片，添加到 resx：



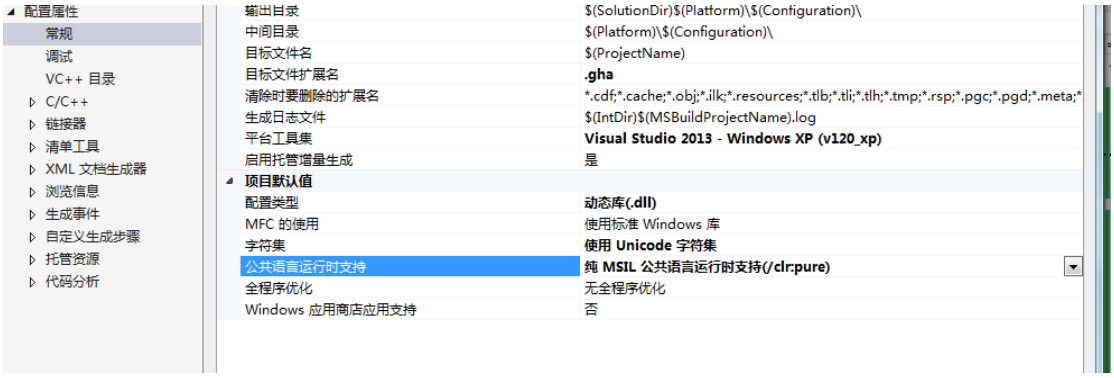


7. 加入图标代码：

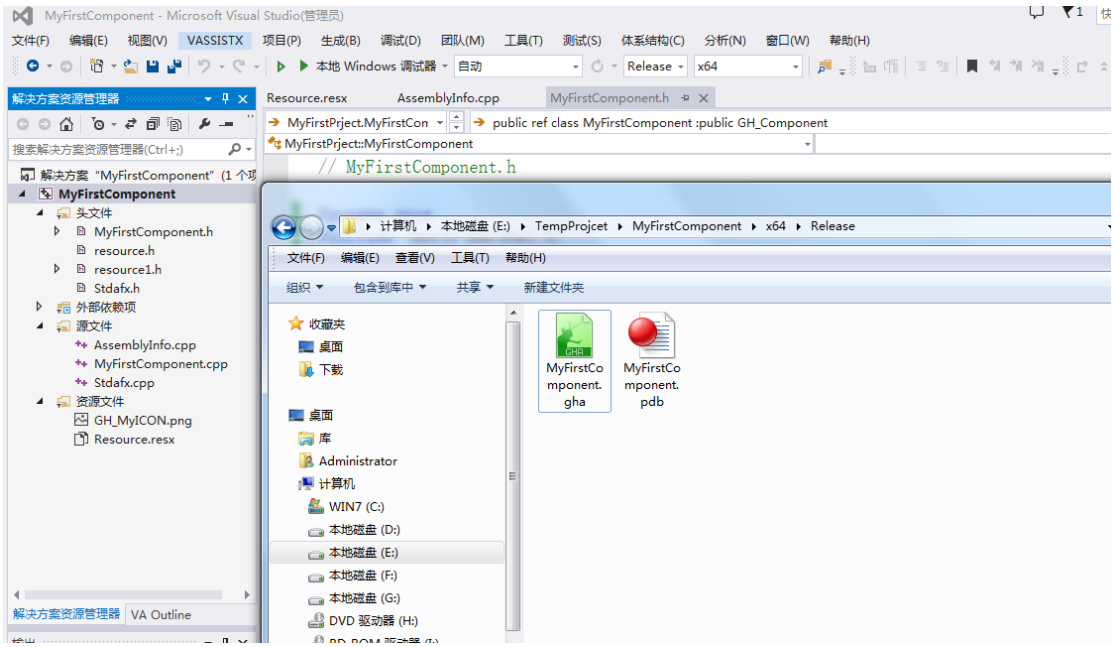
```
        virtual property GH_Exposure Exposure{GH_Exposure get()override{ return GH_Exposure::primary; }}
        virtual property  System::Drawing::Bitmap^Icon{ System::Drawing::Bitmap^get()override
        {
            Resources::ResourceManager^ rm =
                gcnew Resources::ResourceManager(L"MyFirstComponent.Resource", GetType()->Assembly);
            auto img = cli::safe_cast<Drawing::Image^>(rm->GetObject(L"GH_MyICON"));
            Bitmap^ icon = gcnew Bitmap(24, 24, System::Drawing::Imaging::PixelFormat::Format32bppArgb);
            auto g = Graphics::FromImage(icon);
            g->Clear(Color::Transparent);
            g->DrawImage(img, 0, 0, 24, 24);
            delete g;
            return icon;
        }
        }
```

8. 修改扩展名为 gha

9. 更改为“纯 MSIL 公共语言运行时支持(/clr:pure)”



10. 编译生成



11. Copy 到 Grasshopper\Libraries，运行 GH：