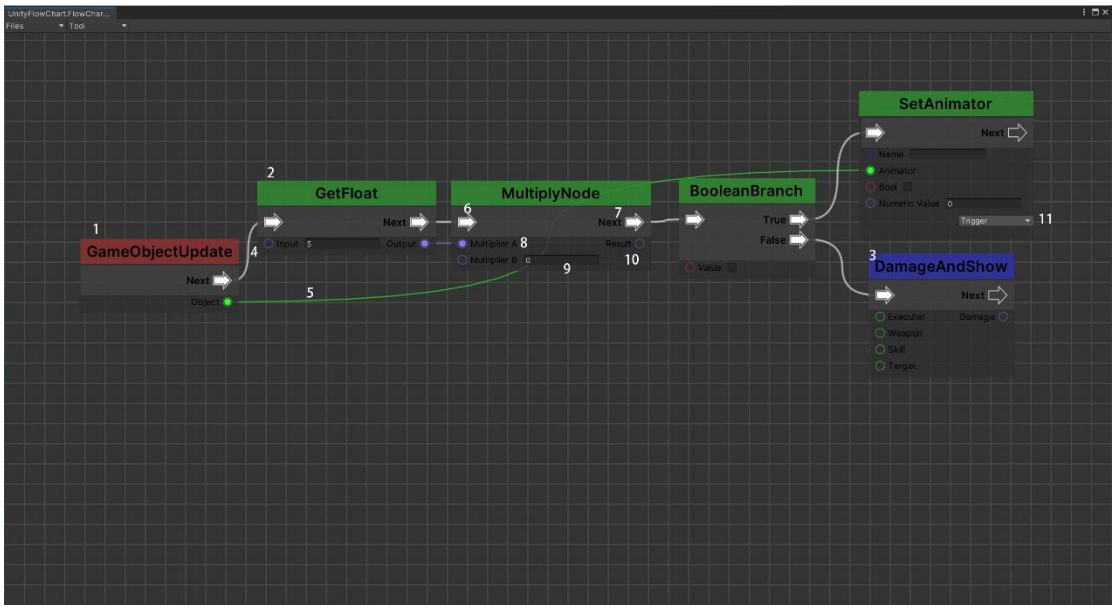
流程图编辑器

目录

概念说明



1、 根节点
2、 普通节点
3、 子图节点
4、 流程连线
5、 参数连线
6、 流程输入点
7、 流程输出点
8、 参数输入点
9、 参数输入框
10、 参数输出点
11、 节点状态选择

打开与保存

可以选中任何一个流程图生成的预制然后右键后选择 Flow Chart 按钮打开该预制，也可以点击空白地方或者通过 Assets/Flow Chart 打开一个空白的流程图。
在编辑器状态通过 File/Save 或者 Ctrl + S 都可以保存编辑过的流程图。如果该流程图是从一个预制中打开的，则会直接覆盖原来的预制，如果是一个新的空白流程图，则会弹出路径选择框选择保存目录。

自定义节点

```
using UnityEngine;
using UnityFlowChart;

namespace Chart
{
    [FlowChartNode("Debug/DebugNode")]
    Unity 脚本|0 个引用
    public class DebugNode : FlowChartNode
    {
        4 个引用
        public enum OutputType
        {
            Log,
            Warning,
            Error
        }
        [FlowChartInput("Debug Data")]
        public string Data;
        [FlowChartStream("Next")]
        public FlowChartNode Next;
        [FlowChartNodeState("Debug Type")]
        public OutputType DebugType;
        52 个引用
        public override FlowChartNode FlowChartContent(System.Collections.Generic.Dictionary<string, object> @param)
        {
            switch (DebugType)
            {
                case OutputType.Log:
                    Debug.Log(Data);
                    break;
                case OutputType.Warning:
                    Debug.LogWarning(Data);
                    break;
                case OutputType.Error:
                    Debug.LogError(Data);
                    break;
                default:
                    break;
            }
            return Next;
        }
    }
}
```

一个继承于 FlowChartNode 的类代表一个节点

FlowChartNodeAttribute

每个节点都必须有该标签
string editorPath: 节点显示在列表中的层级与名称
FlowChartNodeType type: 表示该节点是根节点还是普通节点
bool hidden: 该节点是否会在列表中被隐藏

参数

输入参数与输出参数都必须带有 FlowChartinput / FlowChartOutput 标签，其 string 参数代表该变量显示在编辑器上的名称。

流程输出

流程输出必须带有 FlowChartStream 标签，string 参数是其显示在编辑器上的名称。节点允许有多个输出节点，由 FlowChartNode.FlowChartContent 方法的返回值决定接下来执行哪一个节点。

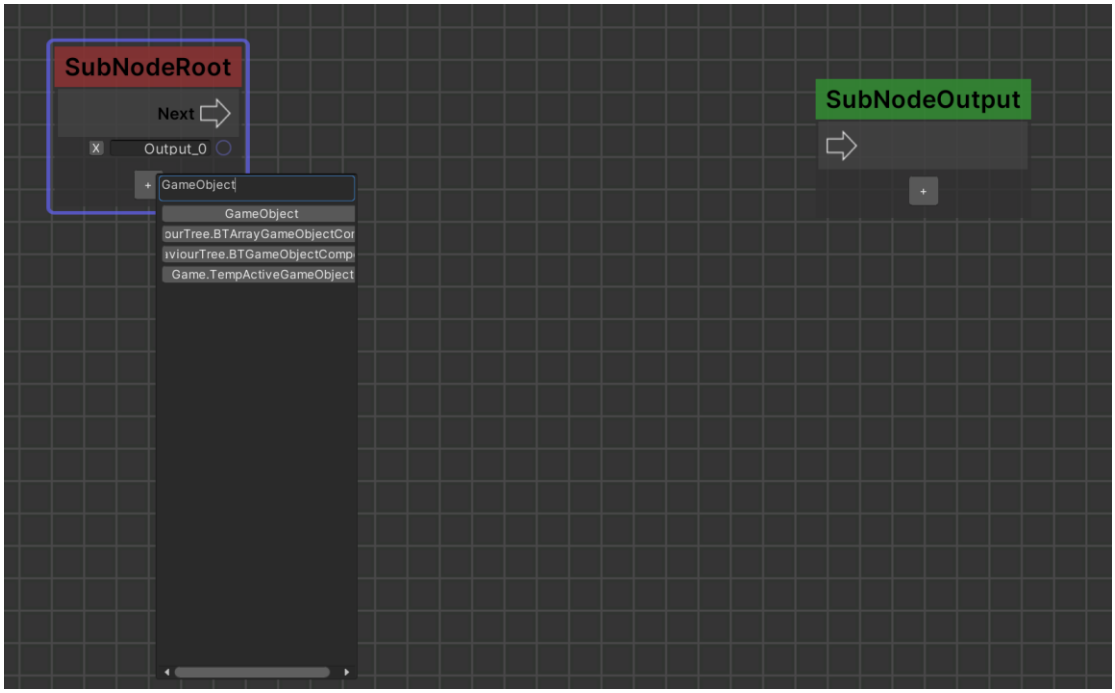
状态选择

状态选择必须带有 FlowChartNodeState 标签，string 参数是其显示在编辑器上的名称。这个字段必须是枚举类型。

抽象方法 FlowChartNode.FlowChartContent

每个节点类都必须实现该方法，该方法用以决定该节点的具体行为，@param 参数是保存了流程图执行中所有参数的字典。
输入参数可以直接在该方法里面使用。*输出参数在方法返回之前必须进行赋值（和 out 参数使用方法类似），否则会导致接下来的节点取不到该参数出现空引用错误。*
节点的返回值决定接下来执行哪一个节点。

子图

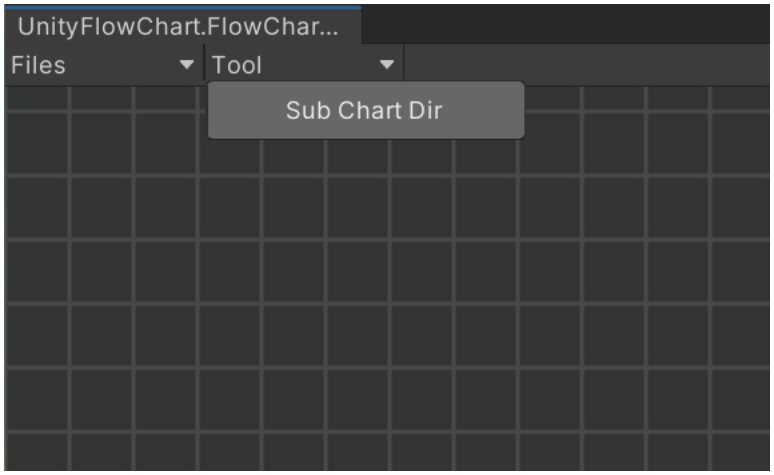


可以把多个节点做成一个子图来实现模块化编辑。子图必须有且仅有一个 SubNodeRoot 节点和一个 SubNodeOutput 节点。这两个节点都可以增加或者删除输入和输出参数。

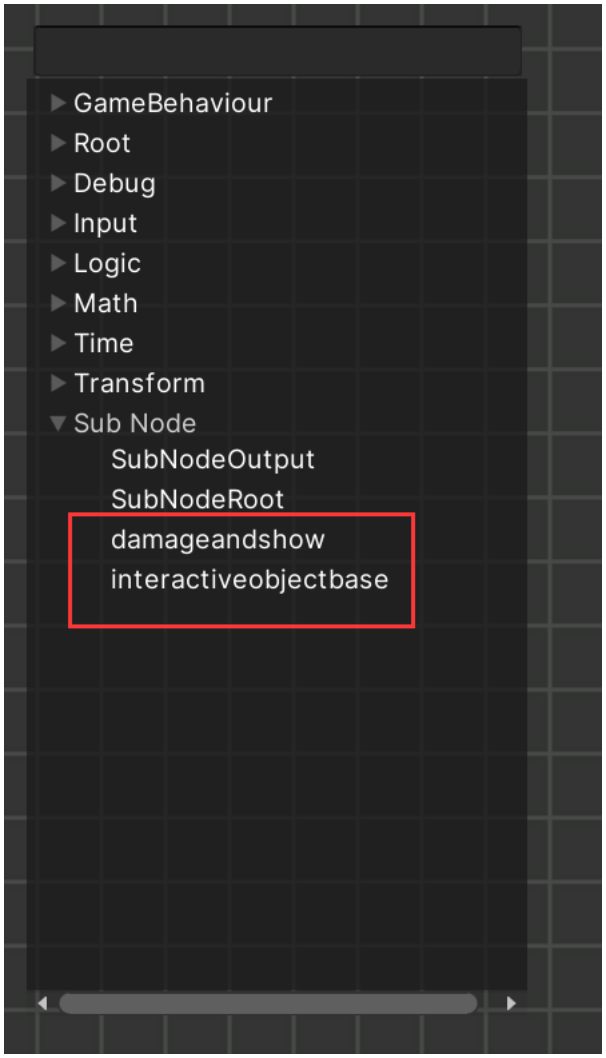
添加删除参数

点击节点中的+号可以添加参数，X 按钮可以删除参数。
添加参数时弹出面板，可以在搜索框中输入参数类型然后点击对应按钮添加参数。
因为类太多，该面板只会在搜索参数过滤到小于 100 个结果后显示按钮。

使用子图



要使用子图必须将编辑好的子图生成的预制保存在规定的目录中（不包含子目录）。该目录可以通过 Tool->Sub Chart Dir 按钮选定。



选定之后可以在节点列表中搜索或者在 Sub Node 子目录里面找到创建的子图节点。

参数输入

参数可以支持在编辑器中直接输入。
目前只支持以下类型进行输入:

1、 基础类型(int, float, double, string, bool).

2、 向量类型(Vector2, Vector2Int 等)

3、 Color, Rect

4、 继承于 UnityEngine.Object 的类型

5、 枚举类型

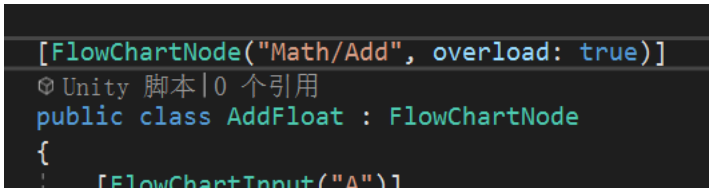
子图节点的任何类型参数目前都不支持参数输入，因为 Unity 无法对任何编译时未确定类型

的参数进行序列化。

参数配对

当进行输入和输出参数连接的时候，必须满足一定规则，比如不能将 int 的输出和 GameObject 的输入进行配对。
进行连接时，只要满足以下的任何一条规则就可以匹配。
1、 相同的类型可以匹配。
2、 任何类型输出都可以匹配 string 类型的输入。
3、 派生类的输出可以匹配基类的输入。
4、 数值类型（int, float, double）与枚举类型之间可以任意匹配。
5、 Vector（2，3，4）类型和 float 可以相互转换。
6、 UnityEngine.GameObject 输出可以匹配任何派生于 UnityEngine.Component 的输入，但是该 Component 必须可以通过 GameObject.GeComponent 获取，否则将会出现空引用错误。
7、 派生于 UnityEngine.Component 之间的类型也可以相互转换，但是也必须满足上一条规则。
8、 派生于 UnityEngine.Component 的类型可以直接转换成 UnityEngine.GameObject 类型。

节点重载



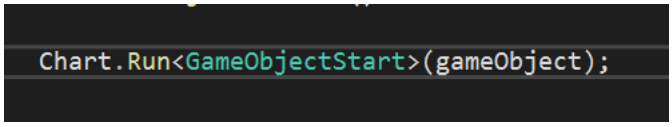
功能：

节点标签中 Path 相同的，overload 参数设置为 True 的节点可以会启用节点重载。启用重载后，给节点连接的*第一个*参数将会作为重载的根据，选择重载中最适合的节点进行更换。（如：给 AddFloat 节点连接上一个 Vector3 输入，节点将会变成 AddVector3，这时候连接第二个参数无论类型都不会改变节点。）

要求：

重载节点必须遵循以下要求：
1、 节点流程输出点、参数输入点、参数输出点数量必须各个相同。
2、 重载节点之间对应的参数（包括输入输出）必须是可以适配的。

触发流程图



获取到流程图编辑器生成的预制，然后调用 GetComponent<FlowChart>().Run，可以运行该流程图，其中的泛型或者 Type 变量表示触发的根节点，如果该流程图中没有这个根节点将不会执行任何操作。该方法的参数是该根节点需要的输出参数。