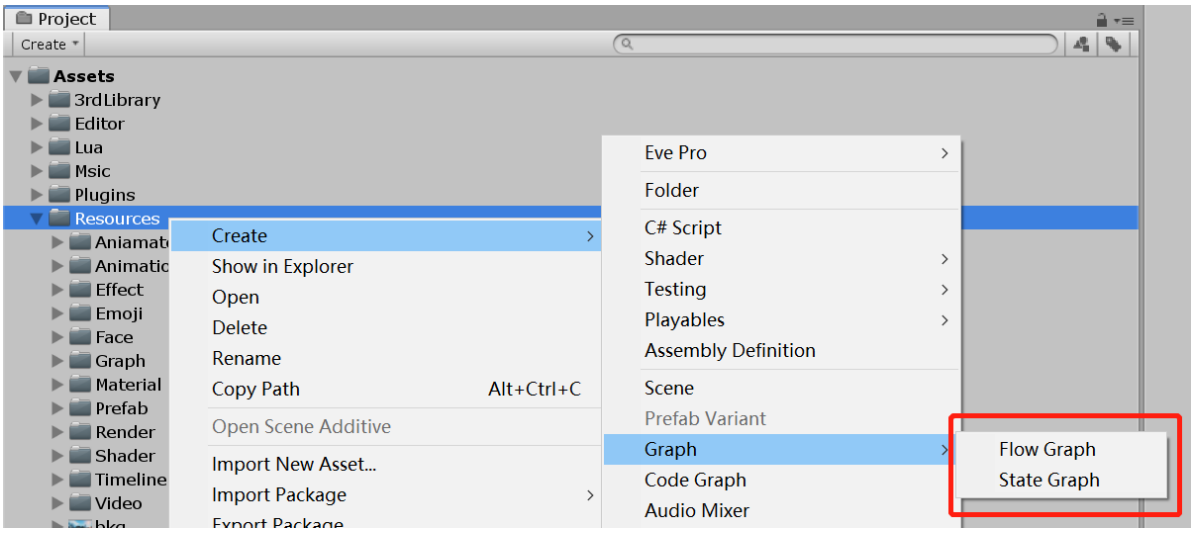
# LogicGraph

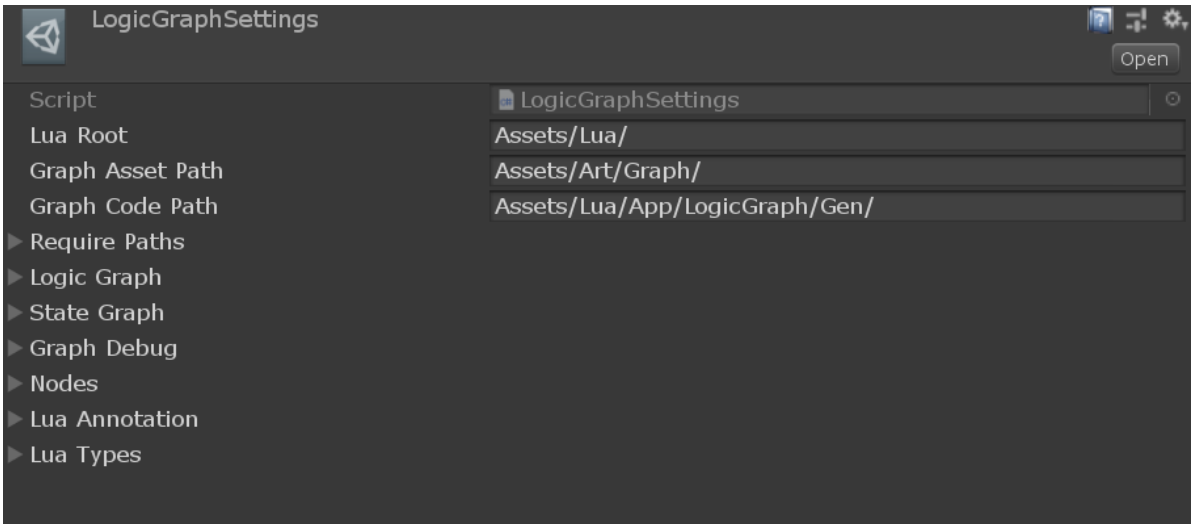
## 简介

### 入口




创建graph之后双击打开asset进行编辑

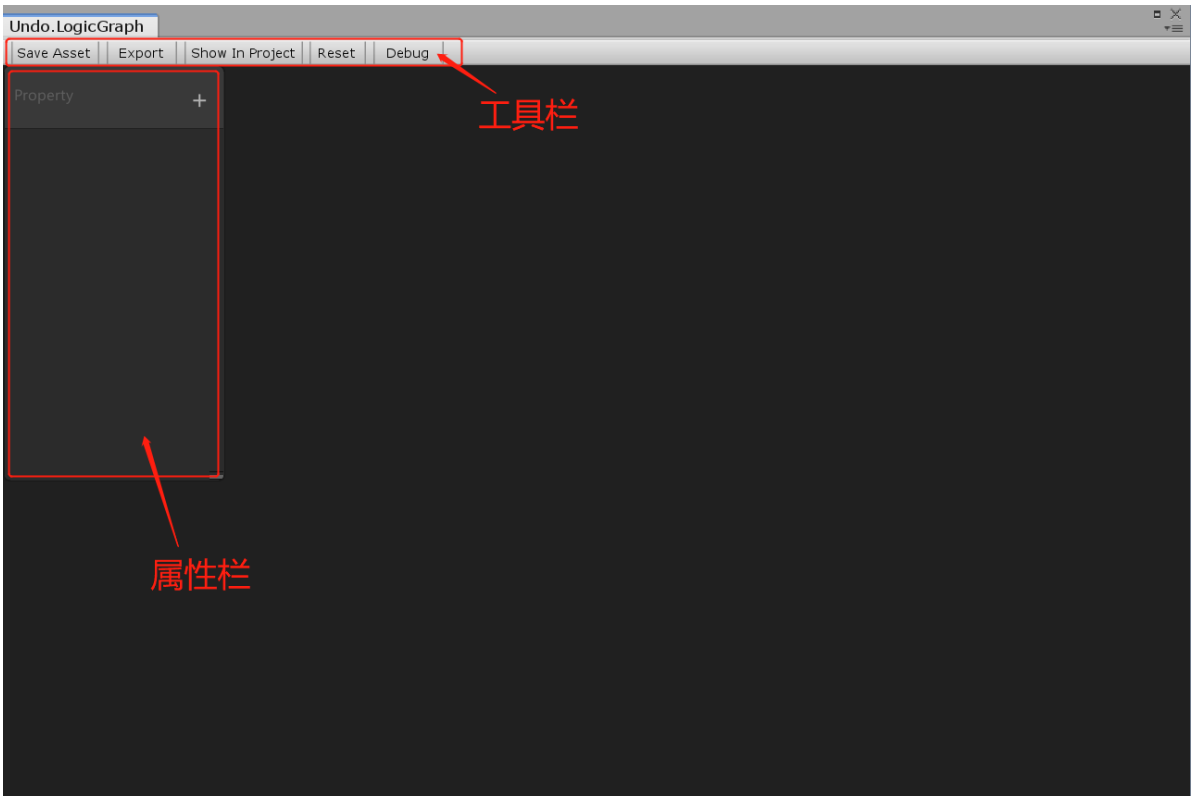
## 设置




设置信息包含以下部分：

- 路径：asset路径，导出的lua文件路径等
- 图关键字：节点类型，节点名，类型名
- 导出信息：导出lua代码时涉及的模板信息
- Debug：调试相关配置

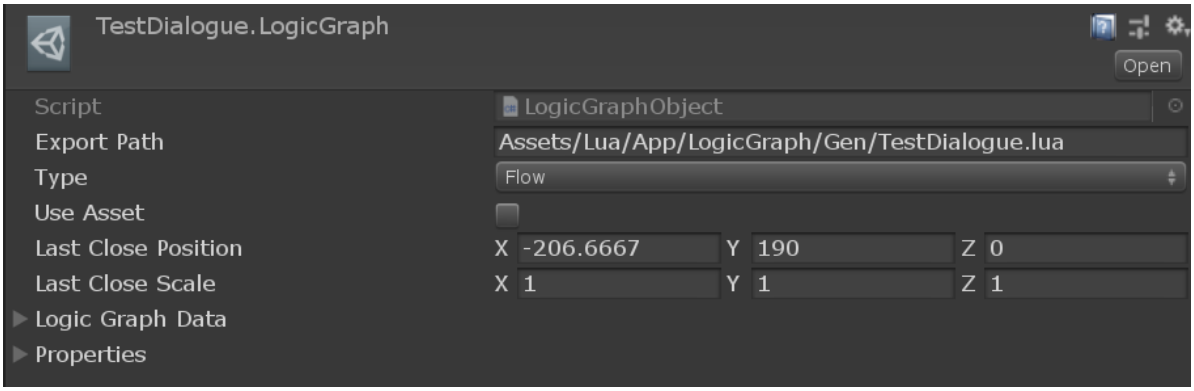
## 窗口




- SaveAsset：保存当前图中的修改
- Export：导出当前图到Lua代码，**（只有在导出Lua代码后，项目才可以使用这张图）**
- Show In Project：在Project窗口显示当前正在编辑的asset
- Reset：重置当前图中的属性以及局部变量，需谨慎使用
- Debug：打开Debug菜单，提供调试相关信息

## 图数据

用于存储图中数据的asset，包含了节点和连线等信息




- ExportPath：Lua脚本的导出路径
- Type：图类型，包括Flow以及State
- UseAsset：勾选后依赖此asset来存储属性，函数调用参数等数据。不勾选时，将在Lua代码中声明这些变量（目前无法在Lua代码中直接声明Object类型）

---

# 基本概念

## 类型

在图中每种数据类型都有一个相应的颜色与之对应，不同颜色的端口之间是无法兼容的（Elua类型除外）

- Int：整型数据，number
- Float：浮点数，number
- String：字符串，string
- Bool：布尔类型，boolean
- Vector2
- Vector3
- Vector4
- Rect
- Bounds
- Curve：AnimationCurve，用于定义曲线
- Color
- Object：继承自UnityEngine.Object的所有类型，包括资源以及各种组件
- Elua：Lua原生类型，在导出时会直接使用该字段进行填充代码段，用于拓展其他类型无法满足需求的情况

## 变量

变量是指在属性面板上创建的变量，这些值作用于整个图的生命周期，并且可以对外暴露，用于在运行图之前的一些定制需求

## 图，状态机

### 图（Flow Graph）

- 定义：代表一个用图连通的指令序列，图的入口是由游戏的运行时状态驱动的（这些状态无法自行定义）
- 使用场景：常用的可以将一个图作为一个表现序列，执行一连串的指令

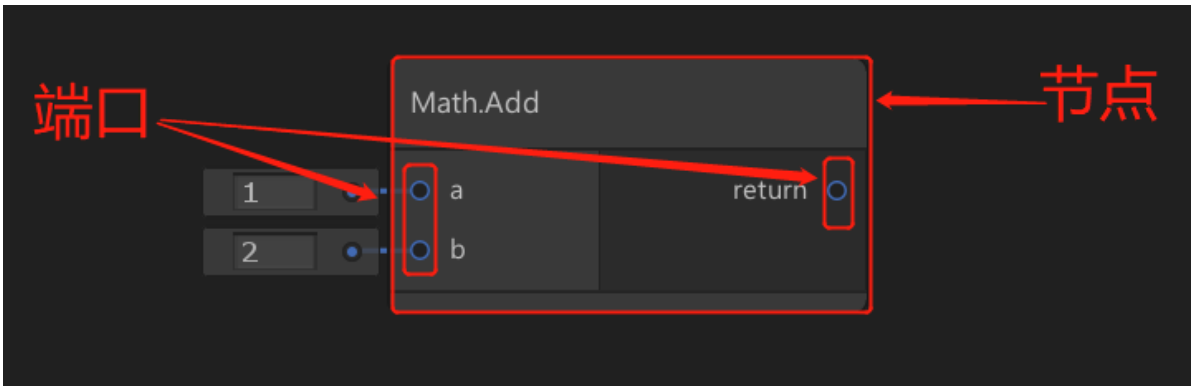
### 状态机（State Graph）

- 定义：一个有限状态机，可以通过事件或其他方法驱动状态机内参数的改变，以达到状态迁移的目的
- 使用场景：逻辑中涉及到玩家输入等由状态驱动的逻辑，例如引导，战斗等内容

---

# 流程图

## 节点




节点是图中的基础单位，节点作为一个最小功能模块，在图运行到该节点时会执行其相应的代码片段

**在创建节点务必遵循EmmyLua注解规范，并且需要设置expose标签**，使用右键图的空白区域，选择菜单CreateNode

```
---@tag expose
---@class LogicGraph.TestClass
local TestClass = DefineClass()

---@param a number
---@param b UnityEngine.Vector2
---@param c UnityEngine.Vector3
---@param d UnityEngine.Vector4
---@param e UnityEngine.Color
---@param f UnityEngine.Rect
---@return number, string
function TestClass:Test1(a,b,c,d,e,f)

end

return TestClass
```

### 节点类型

LogicGraph.TestClass将作为命名空间以及类名，在创建节点时被使用。方法中使用到的参数，以及输出的结果需要添加注解以提供图中节点的端口信息

- Custom：业务逻辑定制的相关节点，后续大部分的节点都将定义在该类别下
- Event：事件节点，目前支持Mono生命周期的全部事件，但是需要配合LogicGraphInstance使用来传递生命周期事件，否则在缺省状态下只支持创建和销毁事件
- Logic：逻辑节点，包含常用的逻辑控制，包括循环，分支等
- Math：数学库
- State：状态机中使用的节点
- Util：工具类型，包含打印调试信息等工具节点

### 端口类型

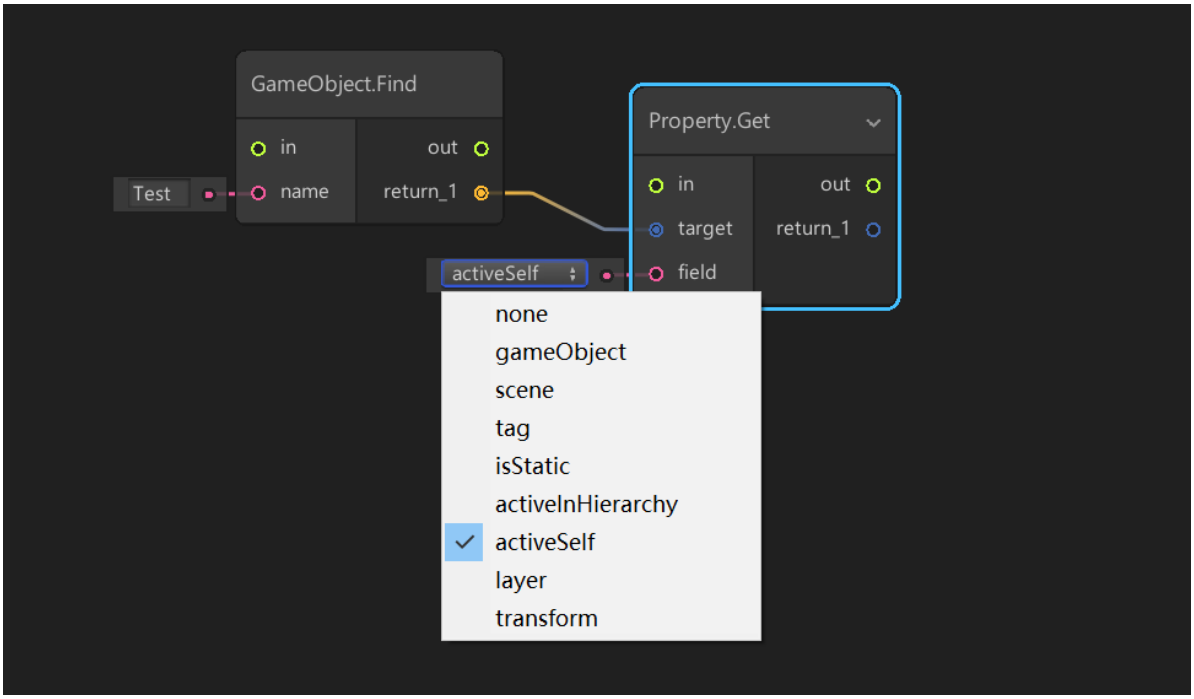
端口是每个节点的必要组成部分，端口的类型通过节点对应的Lua代码确定，端口连线之间将端口名作为id的组成部分，所以端口名（参数名以及返回值）不应该经常变动，并且在变动之后要重新导出图的代码

- 值类型相同：两端的值类型相同
- 端口类型不同：满足从输入节点到输出节点的原则
- 单一输入原则：一个端口可以接受的输入只有一个（状态机中状态节点例外）

## 属性

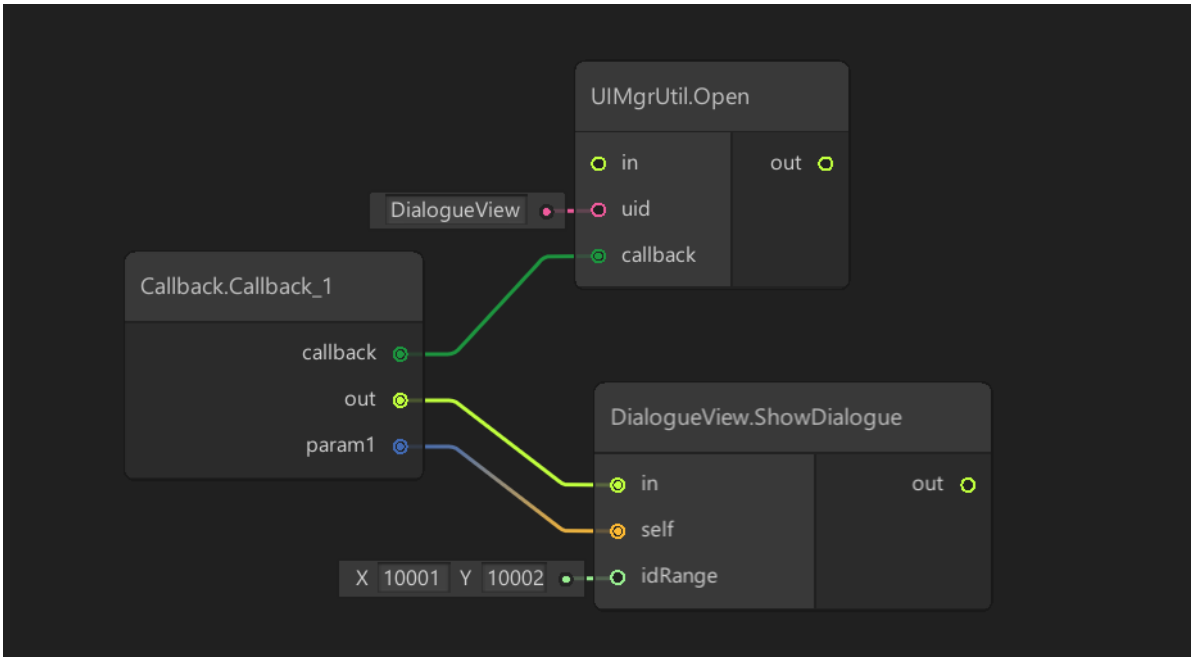
将属性添加至Class定义中，在Property.Get节点中即可获得这些属性，从下拉列表中需要使用的属性名称

```
---@field public transform UnityEngine.Transform
---@field public layer integer
---@field public activeSelf boolean
---@field public activeInHierarchy boolean
---@field public isStatic boolean
---@field public tag string
---@field public scene UnityEngine.SceneManagement.Scene
---@field public gameObject UnityEngine.GameObject
---@class UnityEngine.GameObject: UnityEngine.Object
```



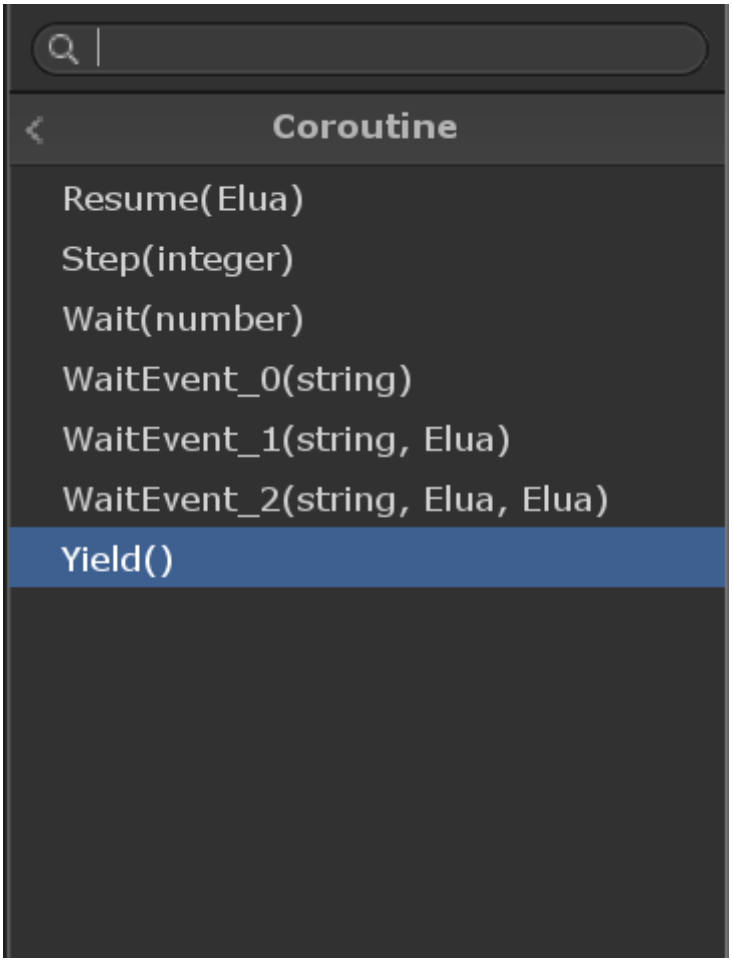

## 回调

异步函数中需使用回调函数来传递结果和执行异步操作，回调函数目前支持0，1，2个参数的类型，需根据需要使用




## 协程

目前入口函数，Start，以及回调函数类型的执行都会通过启用一个协程进行执行。这样做的好处是在协程内可以方便地执行诸如等待或者其他耗时的操作，而无需阻塞。




- Yield：暂停协程
- Resume：恢复协程
- Step：等待n帧，n帧后接着执行
- Wait：等待n秒，n秒后接着执行
- WaitEvent：等待事件，参数为事件名，事件参数1，事件参数2

# 状态机

## 状态节点

对应状态机运行时的一个状态，通常会在进入状态或者离开状态时执行相关节点

- AnyState：从该节点可以迁移至任一节点
- Entry：状态机入口节点，在状态机启动后会首先运行该节点
- Exit：在状态机完成其生命周期时执行相关的回收
- State：状态节点，状态的迁移往往都会对应一个目标状态节点

## 迁移节点

Transition，迁移节点提供State节点之间的迁移，迁移类型包含：

- 相等（==）
- 不等（~=）
- 大于（>）
- 大于等于（>=）
- 小于（<）
- 小于等于（<=）