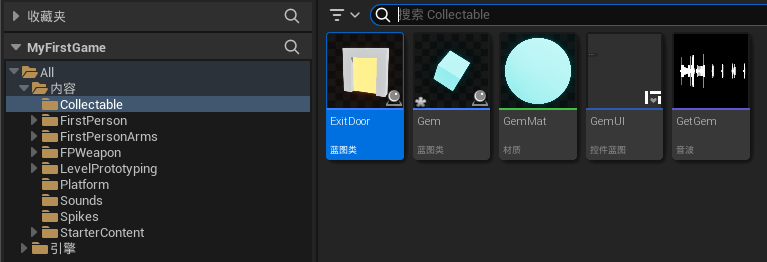
### 宝石创建、销毁、UI显示

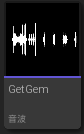
这里，我们将宝石设置为一种用于通关的手段。当所有宝石都收集完成后，通往下一关的关卡的大门会打开。收集完一个宝石之后，会销毁该宝石（而不是销毁所有宝石对象）。屏幕上会显示宝石的总数和已收集的宝石的数量，并判断两者是否相等，若相等，则打开关卡大门。

#### 一、前置素材

首先在资源浏览器中创建Collectable文件夹，后续会创建：ExitDoor/Gem/GemMat/GemUI/GetGem五项内容。



首先在资源浏览器中导入声音材料，声音为收集宝石瞬间发出的声音，并将声音命名为GetGem

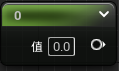


接下来创建Gem的材质GemMat。将Gem设置为蓝色

**Vector Parameter**节点

连接至基础颜色

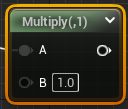
**Constant**节点（常量节点）



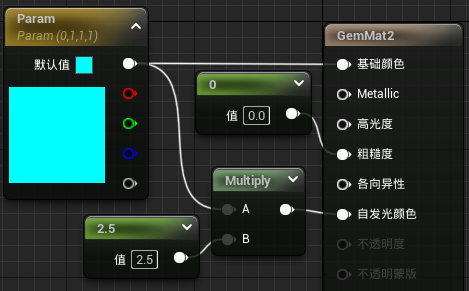
将该节点连接至粗糙度，让材质光滑

为了让材质发出同样颜色的光，从VectorParameter节点连接出

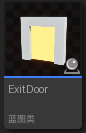
**Multiply**节点



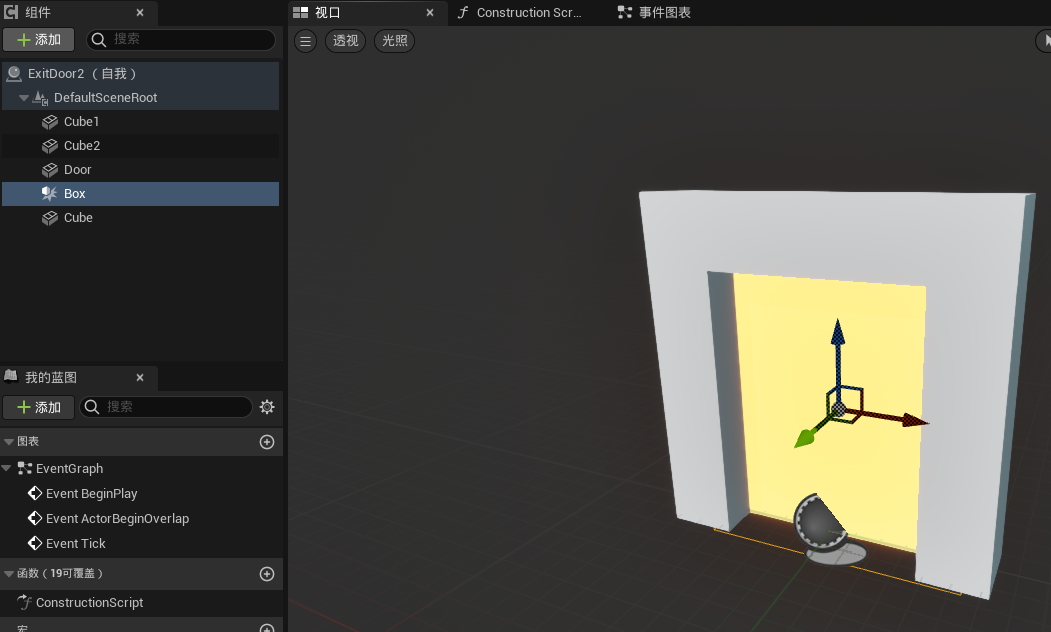
将倍数值设置为常量2.5，并将其连接至自发光颜色



接下来创建关卡大门，资产为蓝图类，父类为Actor，命名为ExitDoor



ExitDoor的属性：三个Cube组件组成门框，一个Cube组件作为大门显示为红色或绿色，一个BoxCollision组件返回玩家与门的碰撞结果。当门关闭的时候，门显示为红色。当门打开的时候，显示为绿色。BoxCollision的大小与door相似。



接下来选中BoxCollision组件，在“细节-事件”中选择“组件开始重叠时（OnComponentBeginOverlap）”

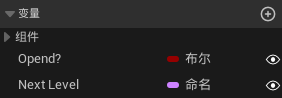
在事件图表中开始编程

在变量中创建两个变量：

布尔类型“Opend？”变量，默认值为false

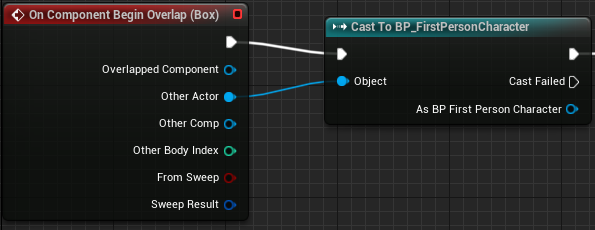
命名类型“Next Level”变量，默认值为none

将变量设置为public



接下来进行节点连接。

首先要确定和BoxCollision碰撞的Obj是BP\_FirstPersonCharacter，连接至OtherActor对象引用

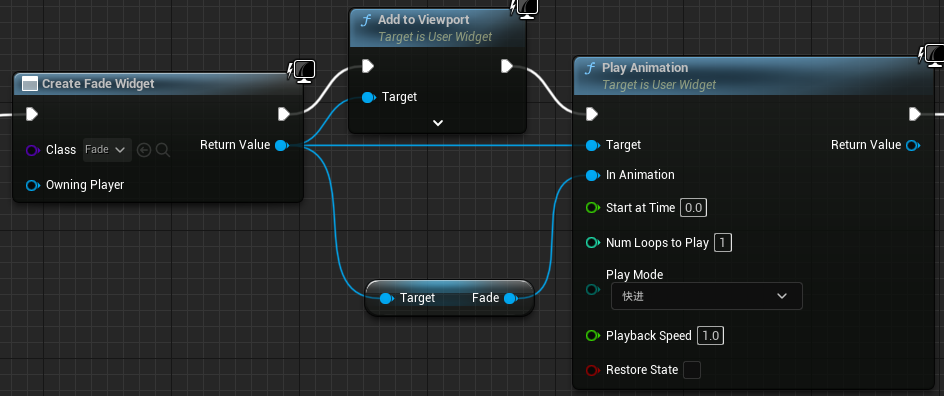


判定碰撞发生后，判断门是不是开的。如果门开了，则：

执行一次fade动画以进入下一关，并进入下一关

首先连接Branch节点，将Condition设置为“Opened?”以判断门是否开启。如果门开启了，则连接DoOnce节点以保证只执行一次

接下来连接fade动画播放节点族

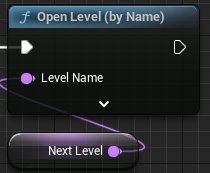


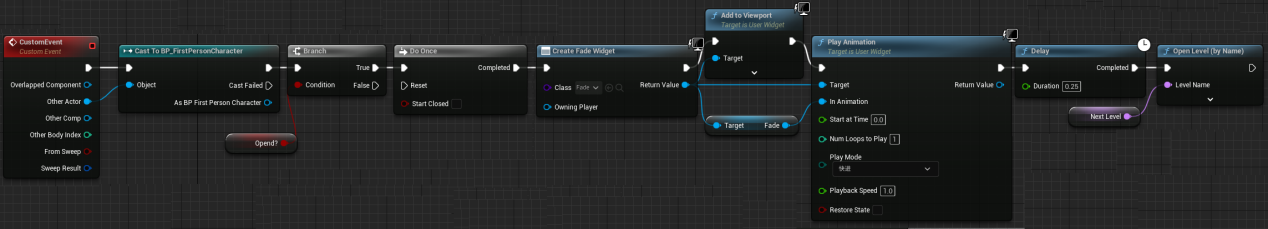
然后进行0.25秒的延迟后，开启下一关卡。即连接Delay节点和Open Level（by name）节点。Level的name为刚才设置的Next Level变量

**Open Level（by name）**节点

//漫游到另一关卡

//目标是GamePlay静态





##### 二、GemUI------WidgetBlueprint

在GemUI中，我们将实现UI显示：屏幕上会显示宝石的总数和已收集的宝石的数量，以及碰到宝石的时候提供的反馈动画。

在 GemUI中创建：

画布面板（canvas panel）

背景模糊（“特殊效果--背景模糊”（background blur））。将模糊强度改为5，在画布面板左上角，用于虚化UI文字背景

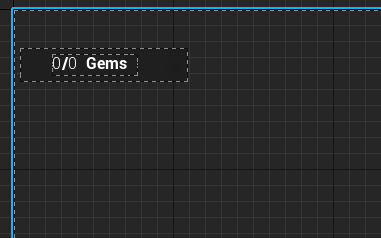
图像1（image）：作为背景模糊的子物件，将图片颜色设置为黑色，不透明度为0.2，用于突出显示UI文字

水平框（horizontal box）：用于显示文本片段

4个文本（Text，来自通用）：用于输入自定义文本。其中两个分别命名为Gems Collected、Gems Needed，将这两个定义为变量（在细节中，第一行），一个命名为/，内容为/，一个命名为Gems text，内容为“ Gems”

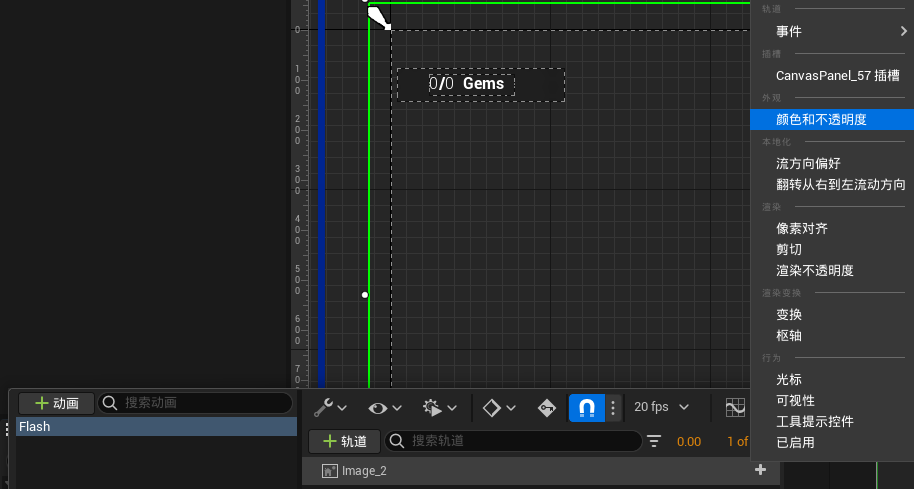
然后把水平框放在背景模糊上面

最终结果如图所示



图像2：作为宝石闪烁动画的展示。将图像大小设置成比画布面板略大，锚点为最右下角的全覆盖，颜色设置为宝石颜色，不透明度设置为0.将该图片的层级拉到背景模糊的上面，以保证闪光出现时不会覆盖UI

用该图像创建动画，命名为Flash，创建“颜色和不透明度”的轨道



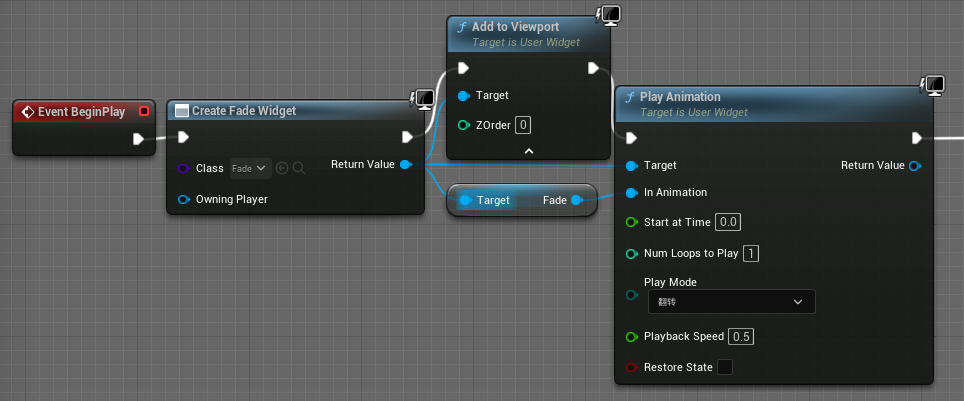
将其展开，在第一帧创建透明度为0的关键帧，在0.1秒时创建透明度为0.2的关键帧，在0.2秒时创建透明度为0.1的关键帧，在0.3秒时创建透明度为0.2的关键帧，在0.75秒时创建透明度为0的关键帧

编译，保存

将上述内容在关卡蓝图中进行操作

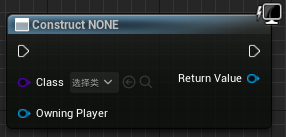
进入关卡蓝图，将GemUI组件加入到关卡蓝图中

在此前，我们的关卡蓝图中应该已经有如图的部分内容

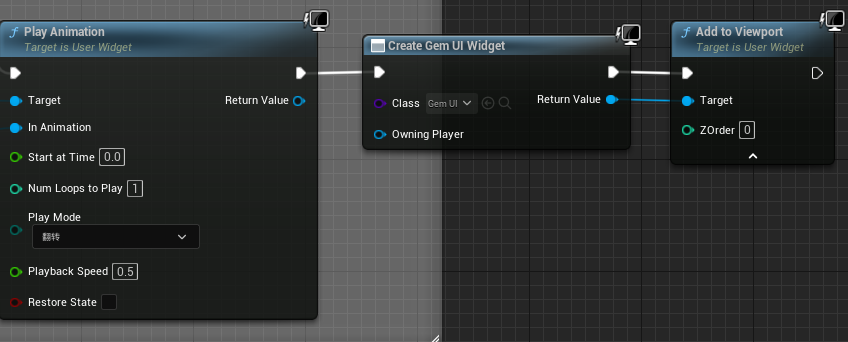


在这部分内容后面添加新内容：

**Create Widget**节点，将类选择为GemUI



**Add to Viewport**节点



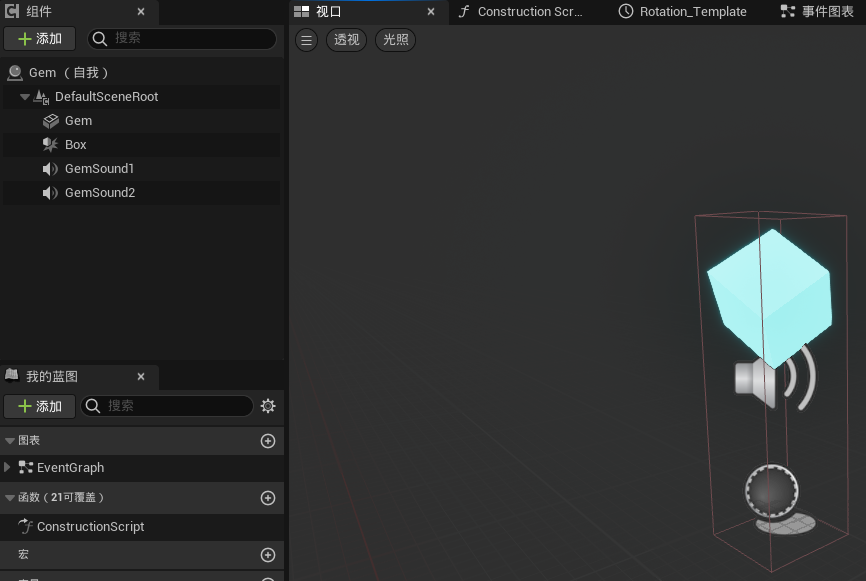
##### 三、Gem------Actor类蓝图

我们需要创建一个Gem对象，来表现宝石的特点。

在组件中添加一个Cube组件，命名为Gem

添加一个BoxCollision

添加两个音频组件，命名为GemSound1、2。如图所示



Cube可以通过三个方向的旋转来达到图片中的效果

同时编辑两个音频组件的细节：

GemSound1和GemSound2：音量乘数分别为4、1；衰减函数为自然声音、空间化方法为双声道；音效分别为GetGem、PressOnPlate；共同调整相应的衰减距离。

接下来进入事件图表（可以先给boxCollision添加“组件开始重叠时”事件节点）

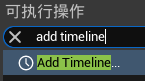
找到EventBeginPlay节点，连接

**Sequence**节点

//按顺序执行一系列的节点



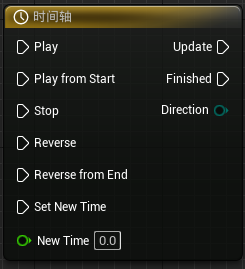
首先执行：Add Timeline节点,命名为Rotation



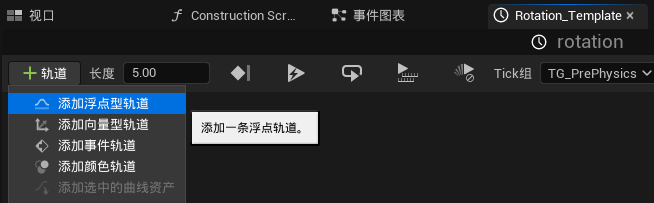
**Add Timeline**节点

//利用时间轴节点可以随时间对值设置关键帧

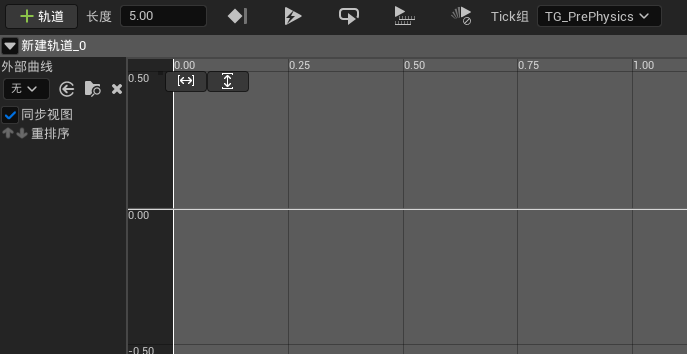
//双击打开时间轴编辑器



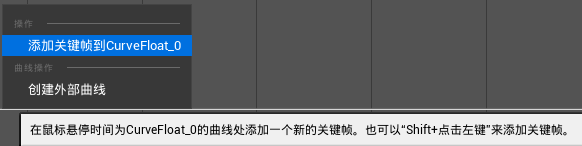
双击打开，如图所示



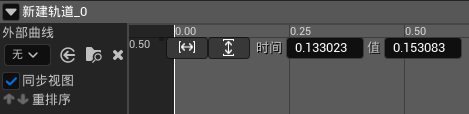
选择添加浮点型轨道



右键选择添加关键帧



此时上方会出现这一帧的时间和值。将两个都修改为0

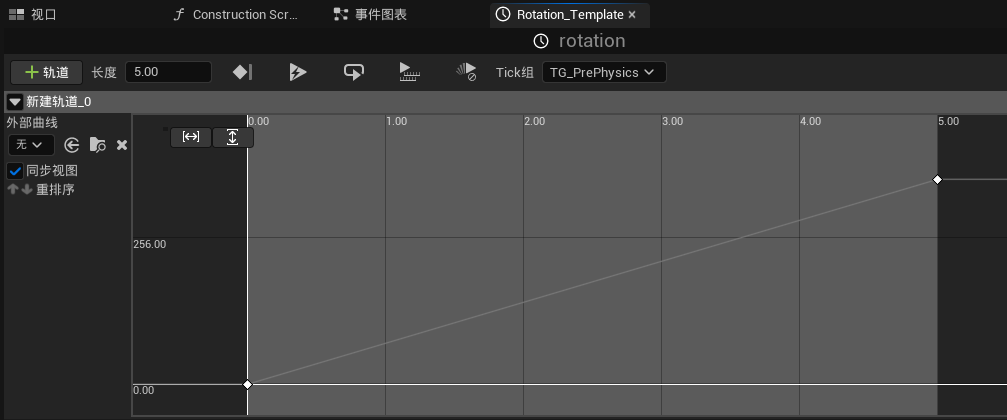


接下来创建第二个关键帧，两个值分别设置为5、360

可以点击如图两个摁键优化视觉展示



最终结果如图所示

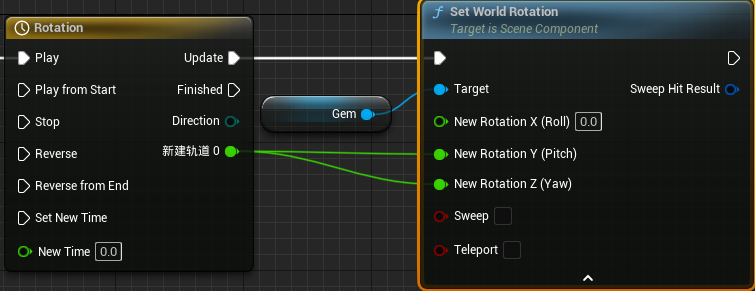


回到事件图表，将时间轴与Set World Rotation节点进行连接，target为Gem

然后将Set World Rotation节点的New Rotation,右键，分割为结构体引脚

将时间轴的新建轨道连接到Y和Z的rotation

然后得到Sequence==0时的节点族



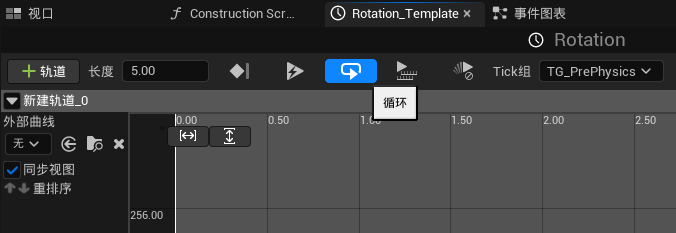
这一节点族用于：随时间的流动，进行WorldRotation的Set

此时在世界中，创建Gem对象，即可看到一个旋转的Cube。此时宝石具有体积碰撞。当我们想消除cube的体积碰撞的时候，我们需要在cube的细节中找到“碰撞--碰撞预设”，选择“NoCollision”

当宝石旋转5秒后会停止旋转。这是时间轴节点未设置好。

返回时间轴节点

打开循环



然后回到Sequence节点，我们进入Sequence==1时的操作

我们需要在一帧的延迟后进行其他操作，即：delay节点，duration参数为0

然后Get All Widgets Of Class

**Get All Widget Of Class**节点

//找到特定类的所有控件

//目标是控件蓝图库

//Cosmetic.This event is only for cosmetic,non-gameplay actions.



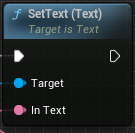
类选择GemUI

接下来是

**SetText（Text）节点**

//直接设置控件文本

//警告：此操作将清除为Text属性创建的所有绑定



然后连接GetAllWidgetOfClass和SetText（Text）之间的部分

接下来的部分比较难理解

从Get All Widget Of Class节点，右下角的Found Widgets：

Found Widgets

用户控件，对象引用数组

所找到匹配过滤的控件

我们需要从这里找出Get（a copy）节点



**Get( a copy )**节点

//给定一个数组和一个索引，返回该索引处数组中的项目的临时副本（因为其为副本，对其进行的修改将不会增殖\回数组中）

//或者可以是下述说法

//给定一个数组和一个索引，返回数组中项的临时副本 在该索引处（由于它是副本，因此不会传播对此项的更改 返回数组）



由虚幻5文档，可以得到：

该标识符指的是（通配符）数组。

此时我们回到GemUI。虽然这些东西之前已经做过了，但是要强调一下，以反映上述内容的原理。

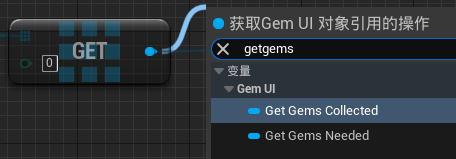
我们需要回到GemUI中，关注层级中两个文本组件：Gem Collected和Gem Needed

我们需要将这两个东西在细节中设置为变量。这样我们可以在其他蓝图中动态改变UI中数字的显示

做完后记得编译保存

然后回到Gem

在Get(a copy)之后，我们需要Get Gems Needed

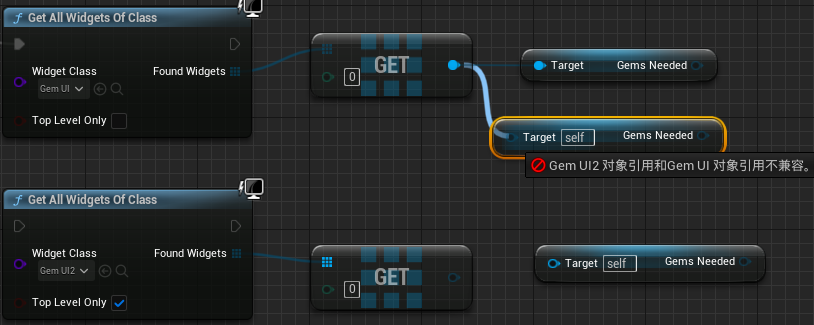




这里注意，有时Get（a copy ）节点不能连接target节点，因为显示：用户控件 对象引用和GemUI对象引用不兼容。



因为按理来说，应当显示（如图内容）



而不是显示用户控件

这个bug是在复刻时发现的

Bug原因至今未能解决

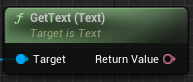
将Target（GemsNeeded）赋值给GetText节点的target

然后将Target节点向后延伸

**GetText(Text)**节点

//获取控件文本

//目标是文本



GetText（text）的返回值是text类型，我们需要将text类型的数据转为int类型

Text不能直接转为int，需要先转为string才能转为int

ToString节点 & ToInt节点



将Gems Needed的文本类型转为int类型后，我们需要将GemsNeeded变量赋给一个初值为1（默认值为0），即

**add**节点



这一节点下面的赋值是0、1、2会产生不同的效果

然后将计算结果转为Text类型

**To Text（Integer）节点**

//基于格式选项将一个传入的整数转换为文本

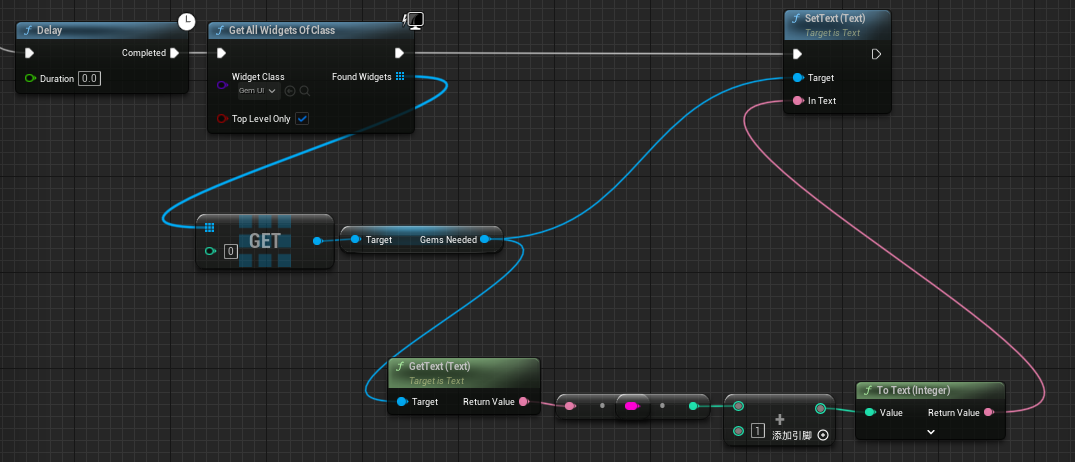
//目标是Kismet文本库



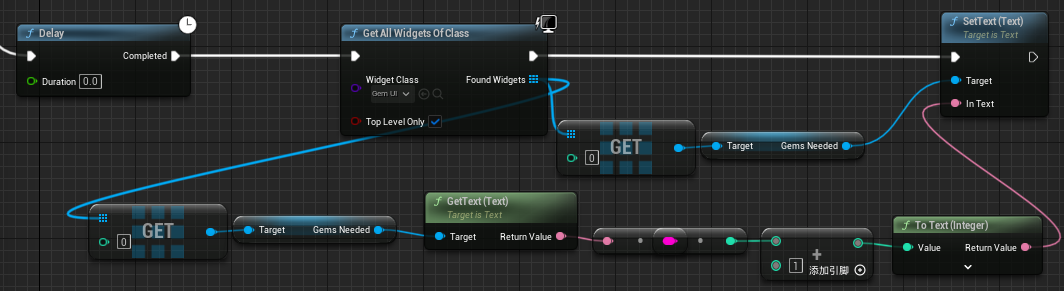
然后将该返回值赋值给SetText节点的In Text参数

事实上，可以直接将add节点的返回值赋值给SetText节点的参数，这样会自己生成ToText（integer）节点

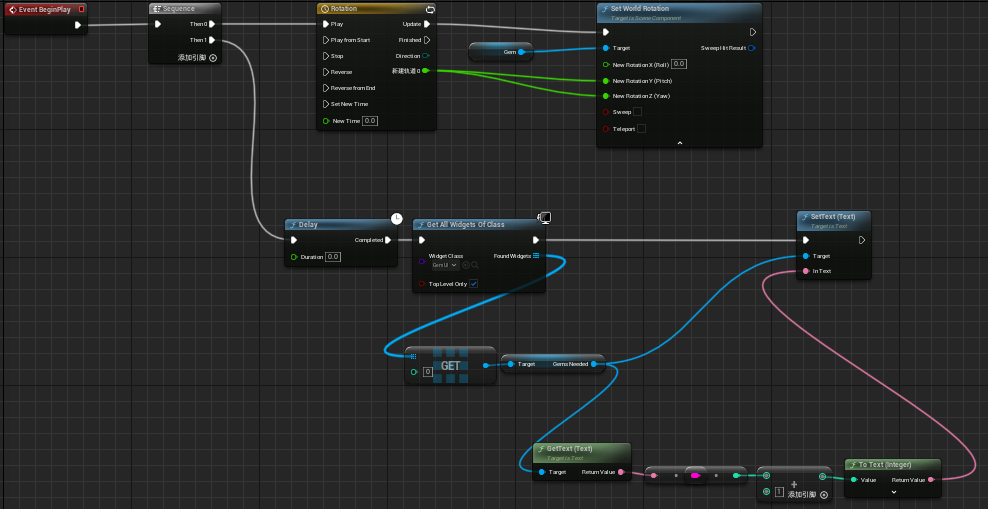
最后得到Sequence==1时的节点族



为便于理解，可以这样表示



最后结果



接下来进行“触碰到宝石即消除”的操作，也就是boxCollision的**OnComponentBeginOverlap**事件触发时

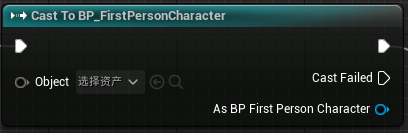


我们需要先将OtherActor强转（或称投射）成BP\_FirstPersonCharacter。即：

**Cast To BP\_FirstPersonCharacter**节点

//尝试将Object作为它可能作为实例的一个蓝图类“BP\_FirstPersonCharacter”来访问

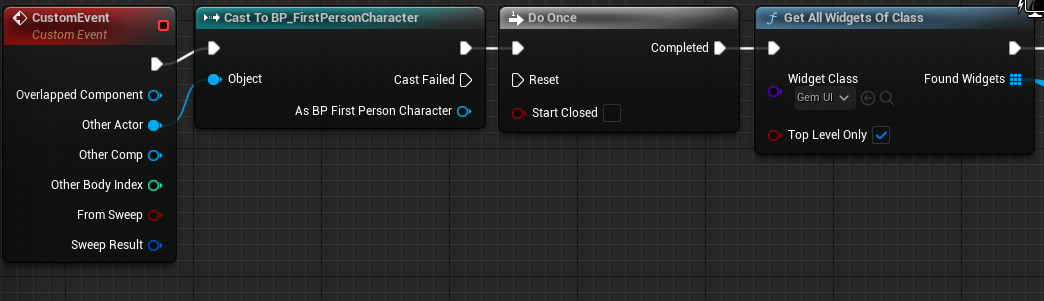
//注意：这将导致蓝图固定被加载，开销可能较高



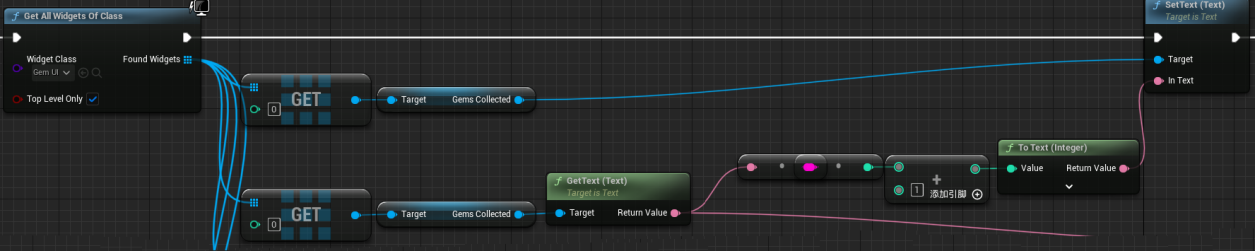
这里不必选择资产，直接将OnComponentBeginOverlap的OtherActor赋值给Cast To节点即可

为了保证碰撞某一宝石事件只发生一次，需要一个DoOnce节点。这样我们也不会意外的从一个宝石中收集到多个宝石。

然后我们需要GetAllWidgetsOfClass，类为GemUI



之后进行SetText（Text），Target和In Text为GemsCollected的文本



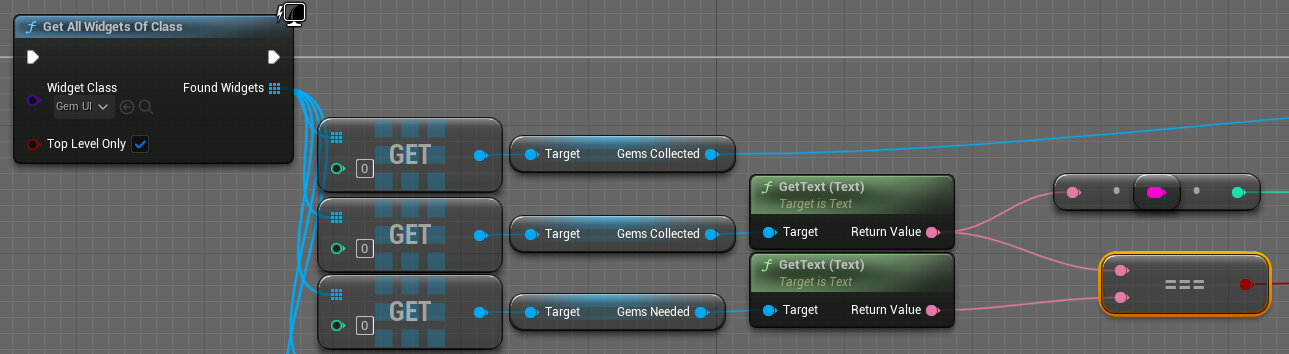
然后Branch，判断GemsNeeded和GemsCollected两个变量的值是否相等

**Equal Exactly（Text）**节点

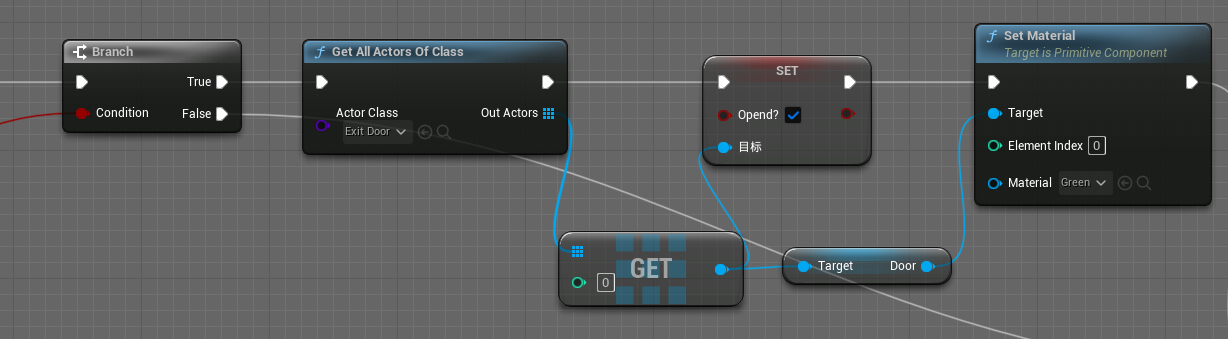
//如果A和B在语言上相等（A==B），则返回true

//目标是Kismet文本库



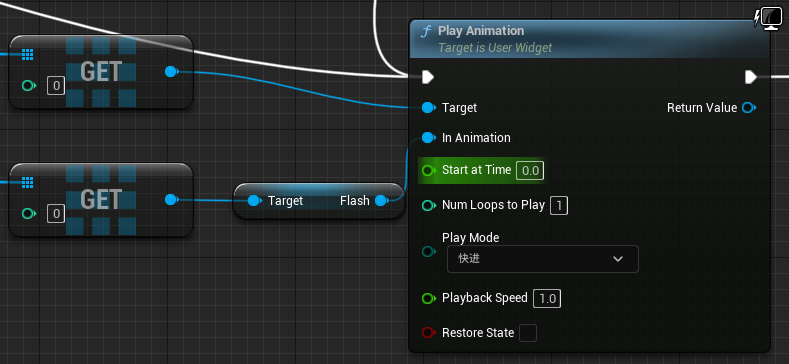


如果相等，就GetAllActorsOfClass，类为Exit Door，然后将门的“Opend？”变量设置为True，将门的材质设置为绿色



这里SET节点的原生命名为“**Set Opend?**”

但无论GemsCollected和GemsNeeded两个变量的值是否相等，我们都应该Play Animation，来把碰到的这个宝石消除掉，播放Flash动画。Target为GemUI的Get（a copy）



然后将组件中的声音组件拉到事件图表中创建节点，接下来我们要播放声音，从声音节点中拉出引线，创建Activate节点

**Activate**节点

//激活SceneComponent，应被本地子类覆盖

//目标时Actor组件



然后将宝石Actor在游戏中隐藏，在5秒的延迟后销毁

问：为什么隐藏了还要销毁

答：不销毁占内存

问：为什么要先隐藏，还要等延迟5秒才销毁

答：如果直接销毁的话，一些相关的行为可能会同时销毁。比如碰撞宝石发出的声音。当没有延迟直接销毁时，会发现声效消失了

即：

**Set Actor Hidden In Game**节点

//将actor设为在游戏中隐藏

//目标是Actor

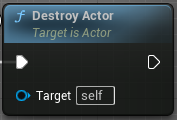


Delay节点,延迟时间为5（按照上面说的）

**Destroy Actor**节点

//销毁actor

//目标是Actor



即完成了Gem的工作

