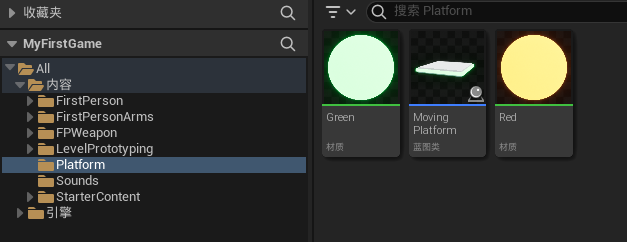
# **创建可移动平台**

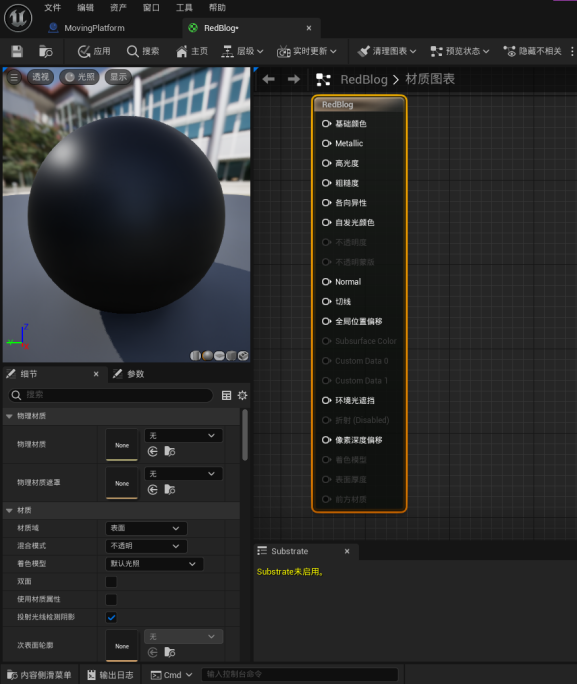
首先在资产管理器中创建Platform文件夹，用于保存可移动平台用到的东西

在platform文件夹中创建两个材质资产一个蓝图资产，材质资产为可发光的Green和Red颜色材质，蓝图为Moving Platform蓝图，Actor类



创建颜色材质（以创建红色为例，绿色同理）

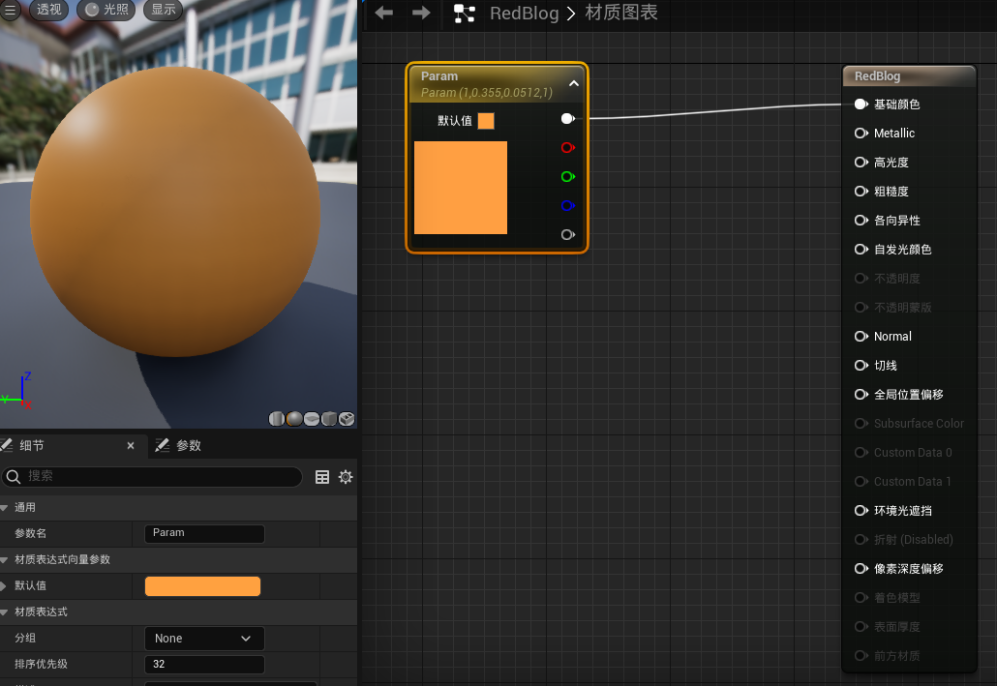
创建好材质后双击打开



创建节点Vector Parameter

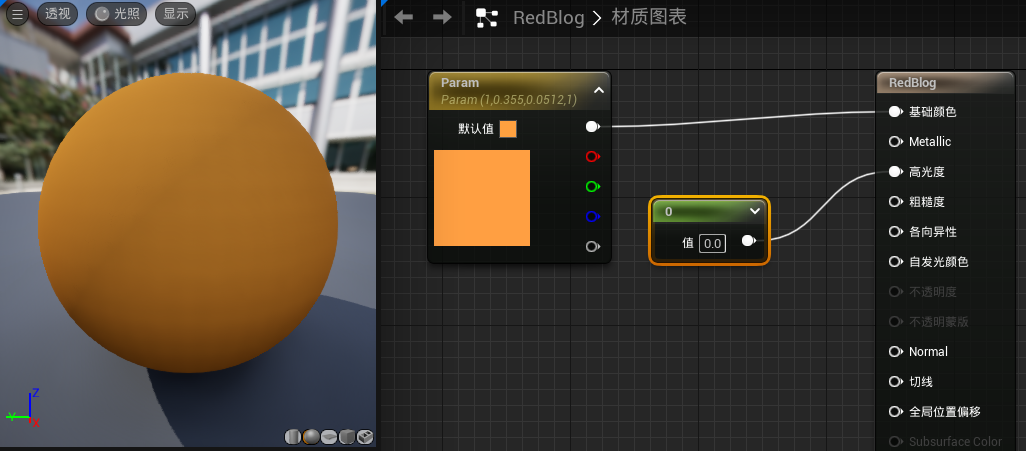


左下角的黑框双击选择颜色，这里选择橙红色，将该节点连接至基础颜色



之后修改高光度使材质反光效果为0

创建**Constant**常量节点，设置其常量值（此处设置为0），并将该值赋值给高光度

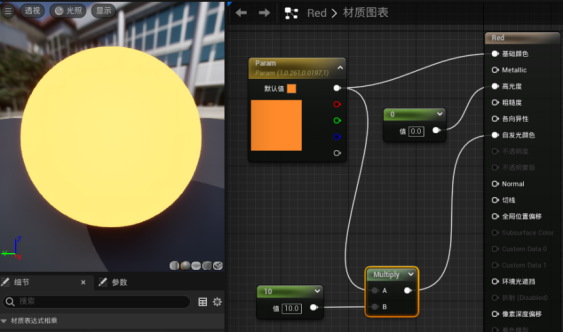


接下来设置发光颜色

创建**Multiply**节点



Mulyiply节点两个输入值，分别为颜色和Constant常量值

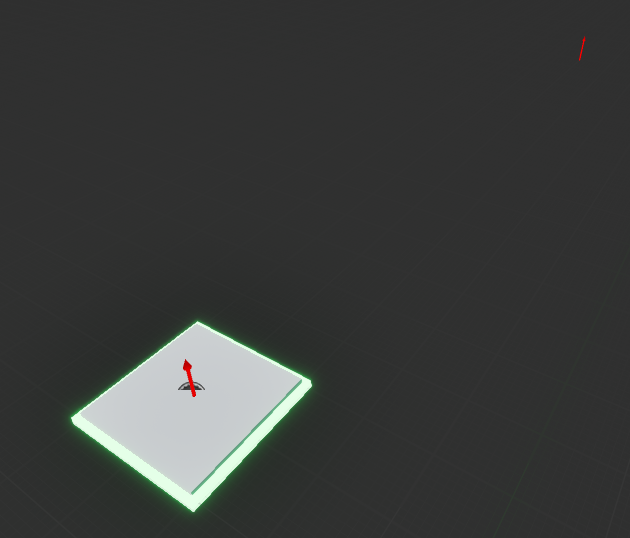


接下来

设置Moving Platform蓝图，使其成为可移动的平台

双击该蓝图，进入视口界面

添加组件，使最终结果为如下所示



添加两个cube组件（搜索关键字不是cube，不知道为何）



两个Cube组件

其中一个命名为Platform，另一个命名为Edge

Edge用来反映平台是否处于Activate状态

若为Activate，Edge发绿光

若不处于Activate，Edge发红光

所以后续的Activate变量默认值为true，Edge默认材质为Green发光材质

添加两个Arrow组件，用于定位platform的起点和终点，命名为Point 1和Point 2（注意让箭头指向上方，好看）



在下面的变量中创建4个变量

Boolean型Ease in变量，默认值false

Float型Move Time变量，默认值3

Float型Pause Time变量，默认值1

Boolean型Activate变量，默认值true

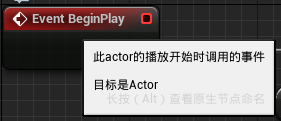
创建完成后将这些变量设置为Public类型，即点击变量右边的眼睛。

之后点击编译和保存，这样变量就可以在细节里设置变量的默认值。

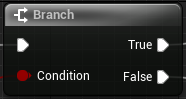


之后进入事件图表界面，创建节点

**Event Begin Play**节点



**Branch**分支节点



**Activate**变量节点--与Branch分支节点链接

在变量里直接拉动Activate变量到事件图表中即可



Branch节点如果为true--链接Set World Location节点

如果为false--链接Set Material节点

**Set World Location**节点

//将此组件放在场景空间中的指定位置，更新相对位置来达到最终的场景位置

//目标是场景组件



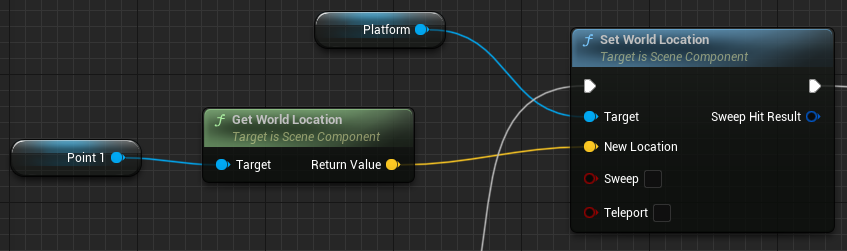
其中：

Target为Platform，Platform节点曲子变量中的组件



New Location为Point 1 的Get World Location

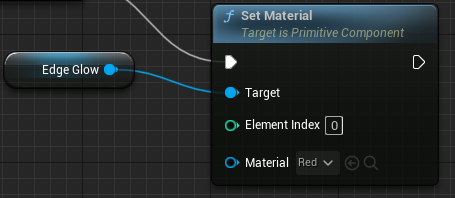
New Location为组件在场景空间中的新位置



**Set Materia**l节点

-让Edge的材质变为Red发光材质

所以Target自然是Edge



在Set World Location之后，需要决定Platform移动的目的地，即

**Move Component**节点

//将一个组件内插到Over Time秒数上特定的相对位置和旋转

//目标是Kismet系统库

//Latent.This node will complete at a later time.Latent nodes can only be placed in event graphs.



其中：

**Component形式参数**为Platform

**Ease In & Ease Out形式参数**，用Ease In变量给他们赋值

Ease In:如为true，插值中将缓入（即缓慢开始）

Ease Out:如为true，插值中将缓出（即缓慢结束）

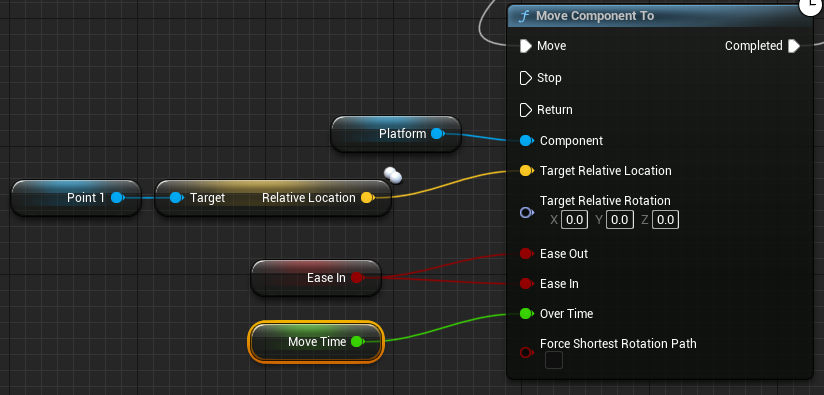
通过Ease In和Ease Out,将实现当目标接近位移重点（即Point 1和Point 2）时，将会缓慢移动

**OverTime形式参数**，用Move Time变量给他们赋值。OverTime反映了插值的时长，反映在从Point1移动到Point2花费的时间

**Target Relative Location形式参数**为Point 1或Point 2的Relative Location，即

**Get Relative Location**节点（场景组件）







上述两个节点图反映了从Point1移动到Point2的情况

我们需要在Point1和Point2之间创建一个循环，使得Platform能够在两点之间移动

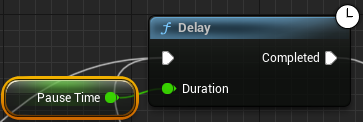
首先要注意到，当Platform移动到当途旅程的终点时，是否会停车一段时间等待其他Actor移动，即

**Delay**流控制节点

//用延迟来执行一项潜在操作（指定为秒）。在倒计时中再次调用并无效果

//目标是Kismet系统库

//Latent.This node will complete at a later time.Latent nodes can only be placed in event graphs.



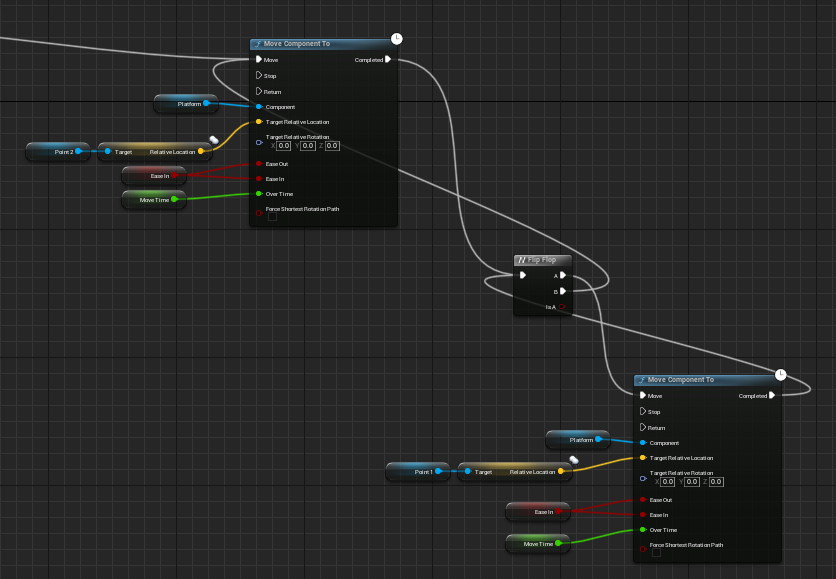
其中的**Durantion形式参数**用PauseTime变量赋值

**FlipFlop**流程控制节点

//在A和B输出之间进行轮替，从A开始

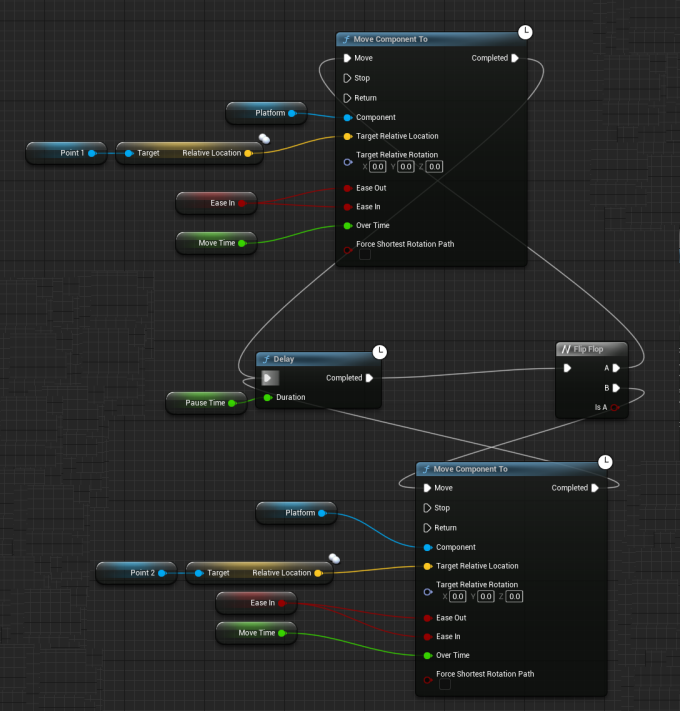


接下来在Point1和Point2之间创建循环链接



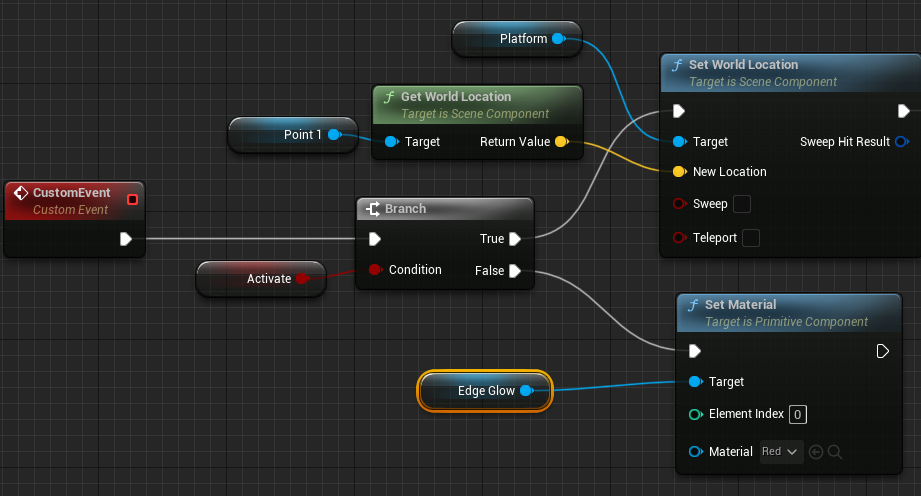
一旦MoveComponentTo完成，便执行FlipFlop，由FlipFlop选择轮替执行哪一个MoveComponentTo

这样并不能使得：当MoveComponentTo结束后Platform会等待Actor，为实现等待效果，应当在每次MoceComponentTo结束后加一个等待节点，即Delay节点



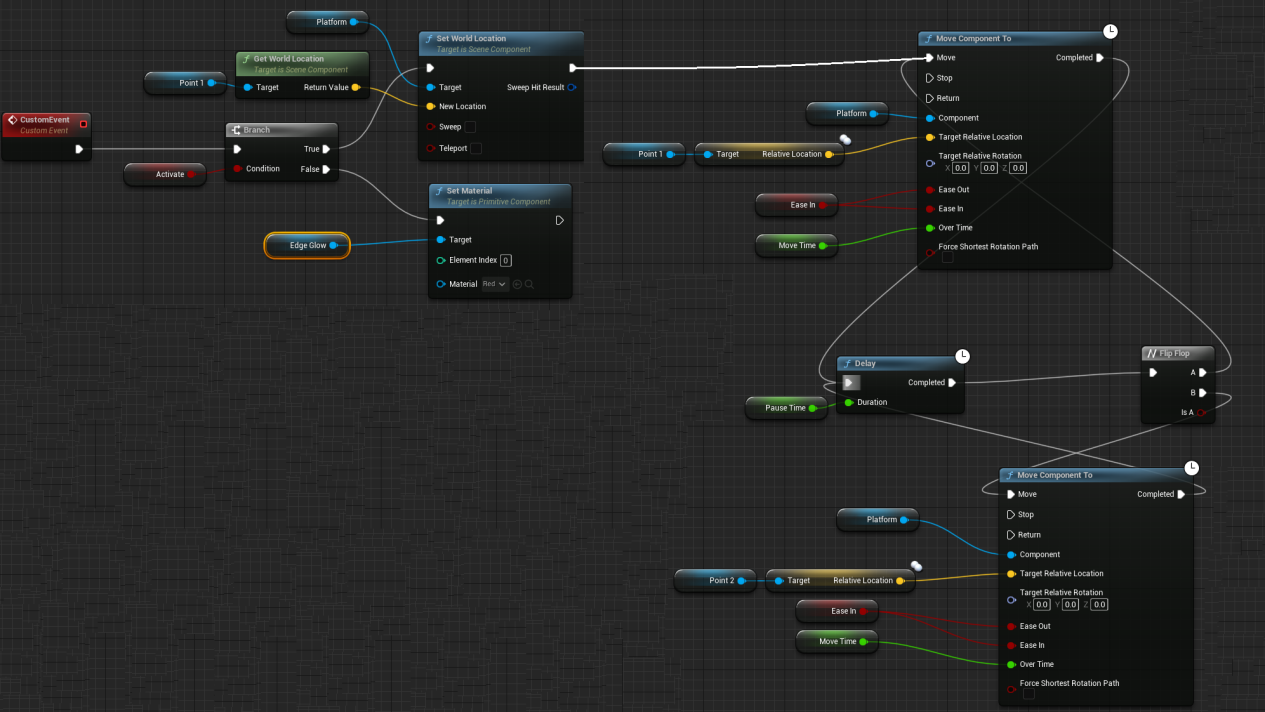
这样就完成了两点间移动的操作实现

再加上前面的判定Activate和Point1&2的部分



//注：左侧的CustomEvent节点应当是**Event BeginPlay节点**

将两部分链接



//注：左侧的CustomEvent节点应当是**Event BeginPlay节点**

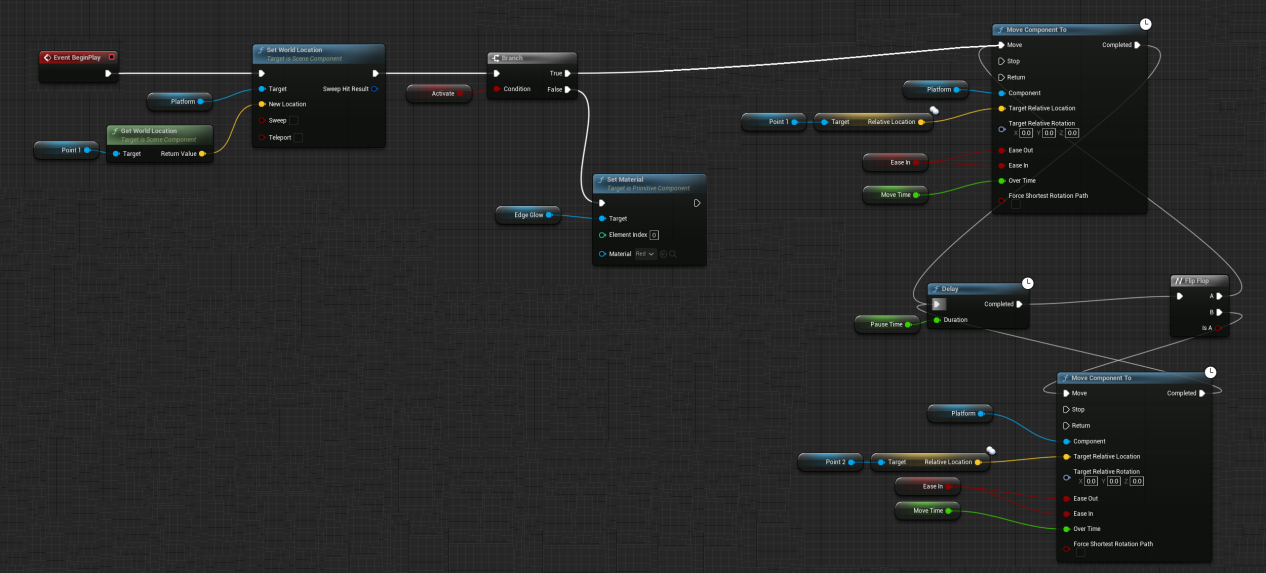
即完成了可自移动平台的创建

这样，当我们在内容浏览器中将MovingPlatform的蓝图直接拉到世界中的时候，就可以创建一个MovingPlatform对象。关注这一对象的两个箭头组件，通过在世界中移动箭头组件，可以决定platform的起点和终点，不用进入蓝图修改箭头组件的相对位置

此时回顾整个蓝图图表，看platform的流程，目前设定的是：从游戏开始进入分支，由分支选择是设置材质或者设置世界位置后开始移动

但是实际上，流程应当是：从游戏开始先初始化platform的位置，然后进入分支决定是设置材质还是开始移动

所以上图有一些流程错误，应当加以改正



# 触发器的实现

该触发器用于实现平台的Activate参数的修改

实现效果为一个压力板

在开始之前，先将可移动平台的Activate变量的默认值设置为false

在Platform文件夹中创建Actor类蓝图PressurePlate并打开

注：该图片是已制作好的Actor蓝图的显示效果

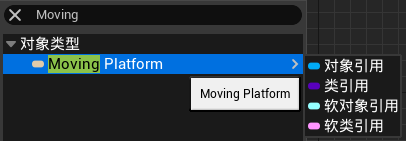
在组件中添加两个Cube组件，命名为Plate和Glow，并将Glow的材质设置为红色发光材质

在组件中添加BoxCollision组件，命名为Box，并将BoxCollision设置为和PressurePlate同样的大小

保存后进入事件图表

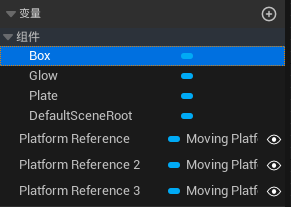
创建变量Platform Reference，变量类型为“移动平台蓝图类型”

搜索Moving Platform,选择对象引用，并将该变量设置为Public类型

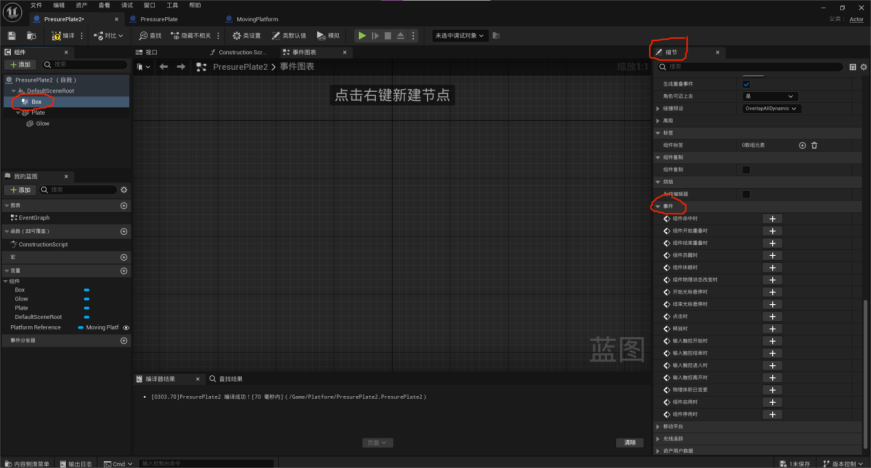


该变量作用是指定一个特定的移动平台。

当你想让一个压力板指定多个特定的移动平台时，创建多个Platform Reference变量



之后点击：box组件----观察他的细节----找到细节中的事件



我们需要给BoxCollision（HitBox）添加相应的事件

选择“组件开始重叠时”



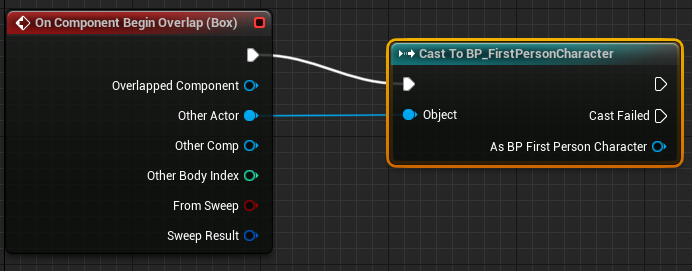
之后事件图表中会出现**OnComponent Begin Overlap(Box)节点**



这时，只要有obj碰到hitbox就会发生事件，但是我们希望只有Character碰到的时候会触发事件

所以我们需要给Other Actor形式参数赋值为我们的FirstPersonCharacter

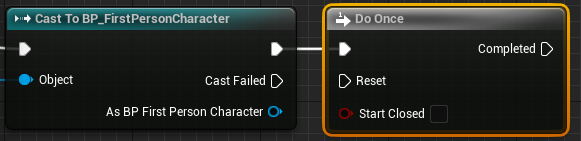
搜索**节点Cast To FirstPersonCharacter**



这个节点负责检查FirstPersonCharacter是否与Box发生了On Component Begin Overlap

**Do Once节点**

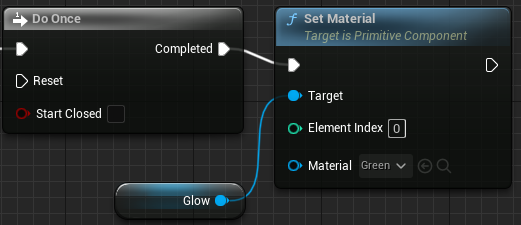
//仅在首次命中时输出触发，但可重设



所以我们不能多次激活这个

一旦Complete，即压力板接受压力后，变换颜色为绿色

创建Glow组建的节点，进行**Set Material**操作



接下来回到Moving Platform蓝图

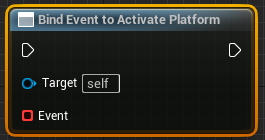
在Moving Platform中创建一个事件分发器EVENT DISPATCHER

命名为Activate Platform

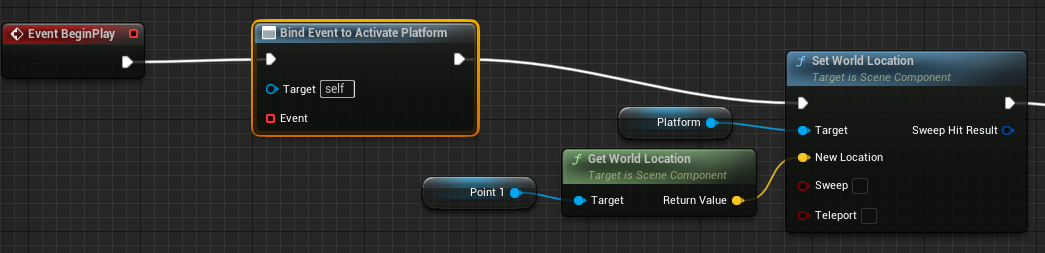


分发器创建完成后，在蓝图中创建Activate Platform节点，属性为绑定(Bind)

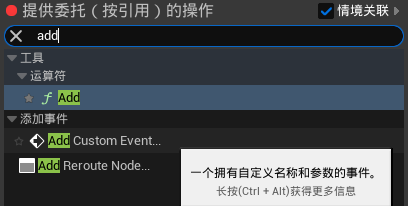
**Bind Event to Activate Platform节点**

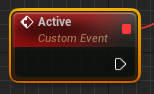


将其安插在Event BeginPlay节点和SetWorldLocation节点中间



从Bind Event to Activate Platform节点的前置参数Event拉出，创建**“自定义事件（add custom event）”节点**,将其命名为Active

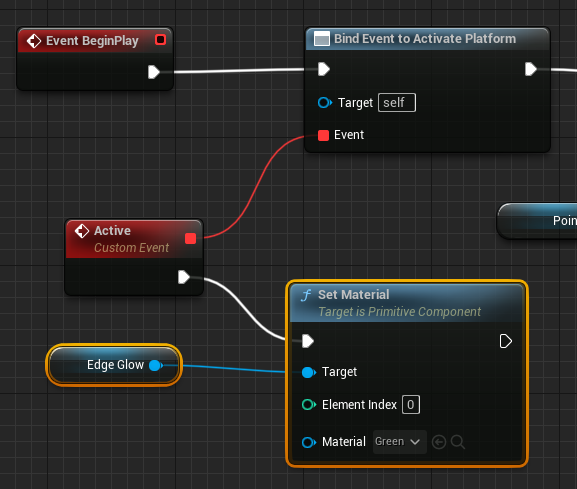


//一个拥有自定义名称和参数的事件

作用：So what this is doing， is this is creating this custom event here that we can call from other blueprints so that we can activate So what we're gonna do is as soon as the pressure plate has been pressed , It's going to call this special event in our moving platform blueprint here . That's going to actually make our platform start moving . While moving platform blueprint. Let's set up this event and what exactly it will do.

翻译：这是在做什么，这是在这里创建这个自定义事件我们可以从其他蓝图中调用，这样我们就可以激活，我们要做的是，一旦压板被压下，它会在我们的移动平台蓝图中调用这个特殊事件。这会让我们的平台开始移动。同时移动平台蓝图。让我们来设置这个事件，看看它到底会做什么。

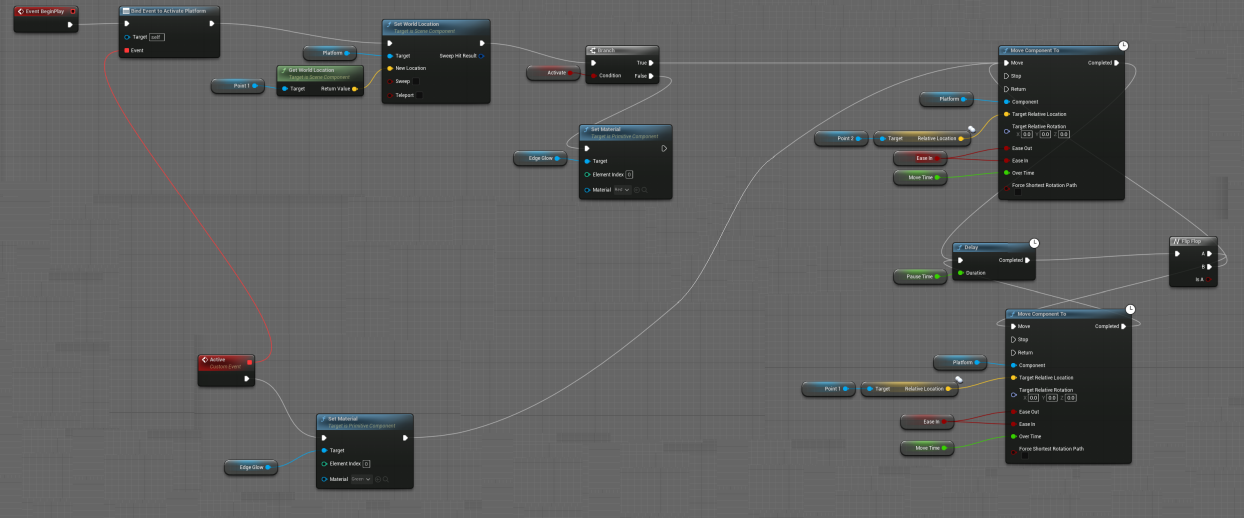
所以一旦他被激活，我们需要设置我们的边缘（EdgeGlow）发光的颜色



我们将材质设置为绿色，因为默认情况下，平台是非活动的（Activate变量默认值为false），这时时红色的。当触发时，会将Activate变为true，也就是变为绿色

在Set Material后，便开始Move Component To

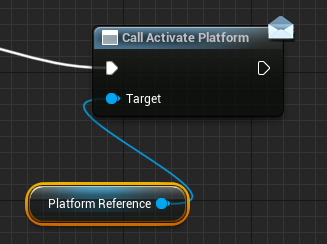
即：将该SetMaterial的右接口连接至后面的第一步执行的Move Component To节点上



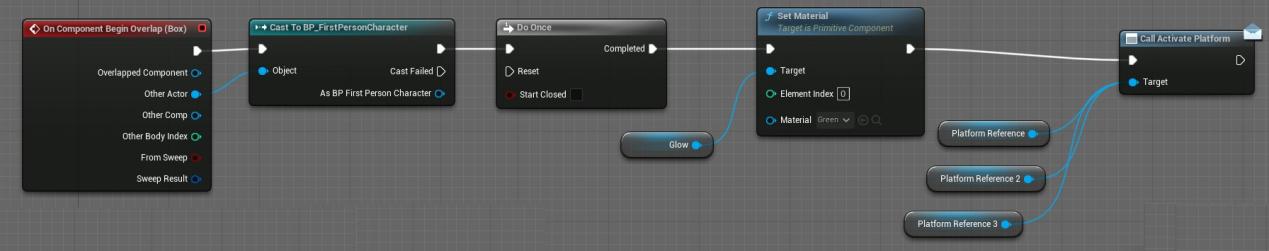
然后我们回到压力板PressurePlate

创建PlatformReference的Get节点，并从该变量节点引申出

Call Activate Platform节点



将该节点与前置节点相连接，即完成了PressurePlate测蓝图创建



在回到世界界面后，我们需要给PlatformReference变量赋值

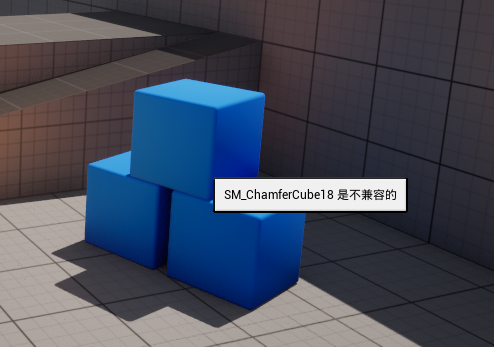
点击压力板，在“细节”中找到“默认”



此时我们需要给三个变量赋值

点击取值器（胶头滴管形状，如下图所示），随后鼠标指针会变为胶头滴管形状。但是只有鼠标指针在指向Platform的时候才会显示为胶头滴管形状，如果指向的不是Platform，则显示为红叉，并给出提示：当前对象类型是不兼容的



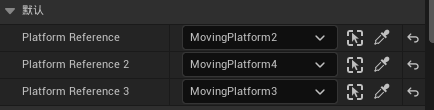




//上述两图截图问题导致的没有鼠标指针的显示

此时将所有的Platform Reference变量都给他们进行赋值即可

注意，一个Platform只能给一个Reference变量赋值。他们是一一对应的



此时即完成了：

当玩家踩到压力板的时候，会触发事件使得Platform的Activate的变量值从false变为true

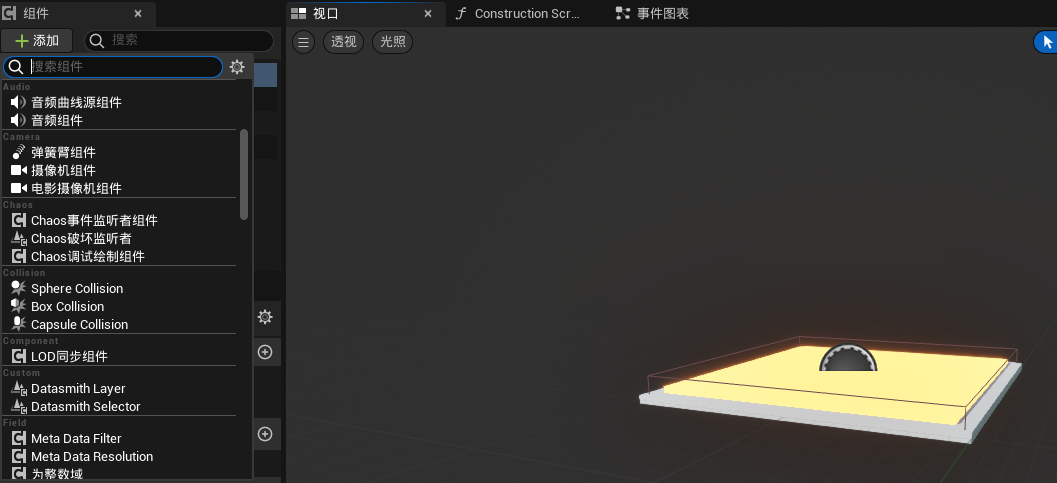
# 在压力板和平台上添加声音

首先，将压力板声音和平台移动声音加入到内容文件夹

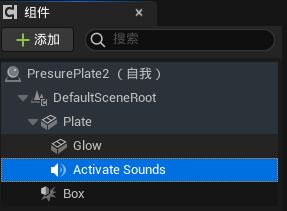


首先开始压力板声音的设置

进入到压力板的蓝图，进入“视口”界面，创建“音频组件”



命名为Activate Sounds，并把该音频组件归为Plate的子组件



之后在该音频组件的细节里，选择要使用的音频文件



并在下面的“衰减”里选择“重载衰减”

将衰减函数改为“自然声音”，调整衰减的内部半径和衰减距离

将衰减的空间化方法改为“双声道”



继续向下在细节中寻找，找到“激活-自动启用”，将默认值改为false



这样我们就可以自行激活声音。

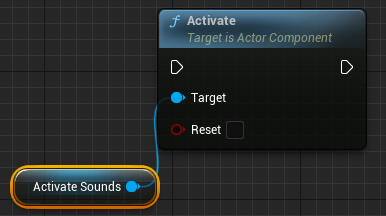
然后回到事件图表，创建Activate Sound的节点，并从Activate Sound节点引申出Activate节点



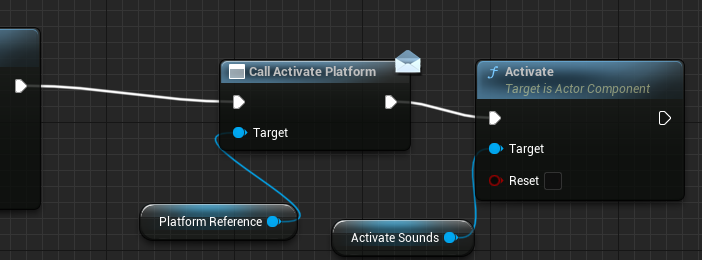
**Activate节点**

//激活Scene Component，应被本地子类覆盖

//目标是Actor组件



然后将该事件接入总事件



此时即完成了对压力板的声音绑定。

接下来进行移动平台的声音绑定

进入移动平台的蓝图，和压力板一样，进入“视口”界面，在组件中添加音频组件，命名为Hum，将该音频组件归为Platform的子组件（归为子组件的好处是发声源会随着Platform的移动而移动）

在细节中选择要绑定的音效，开启重载衰减，修改衰减函数为“自然声音”，调整内部半径和衰减距离，将衰减的空间话方法改为“双声道”

然后就可以保存了，不需要将“激活-自动启用”的默认值改为false，也就是这个声音会一直响，若想让他在平台unActivate的时候不响，将音量设置为0即可

然后回到事件图表界面开始安置音频效果

此时我们需要注意我们的平台，是处于unActivate状态还是处于Activate状态

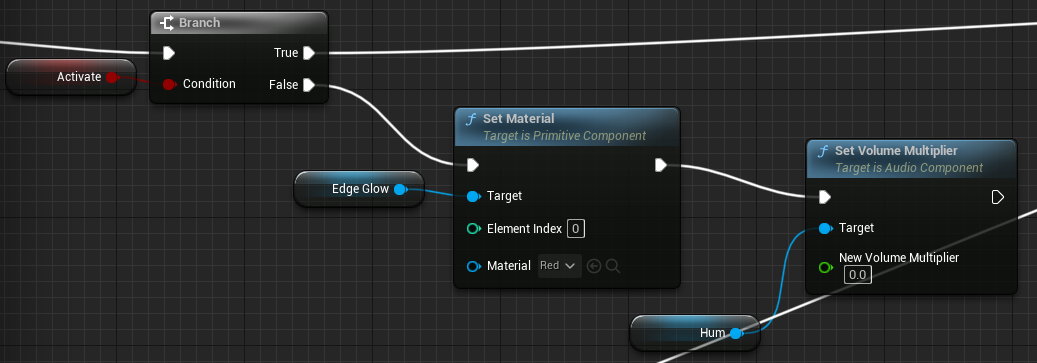
如果处于unActivate状态，需要把我们的Hum音频组件的音量调整至0

**SetVolumeMultiplier节点**

//设置一个新的音量乘数

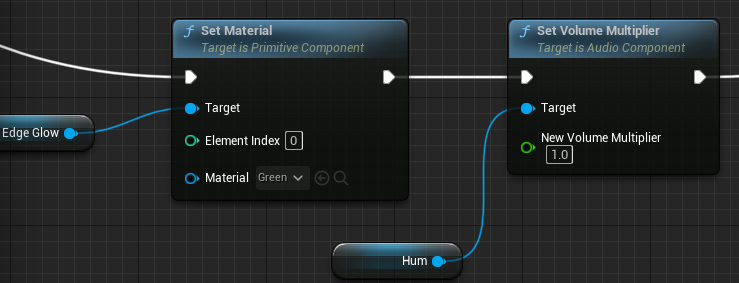
//目标是音频组件





同样的，当处于Activate状态时，将Volume设置为1

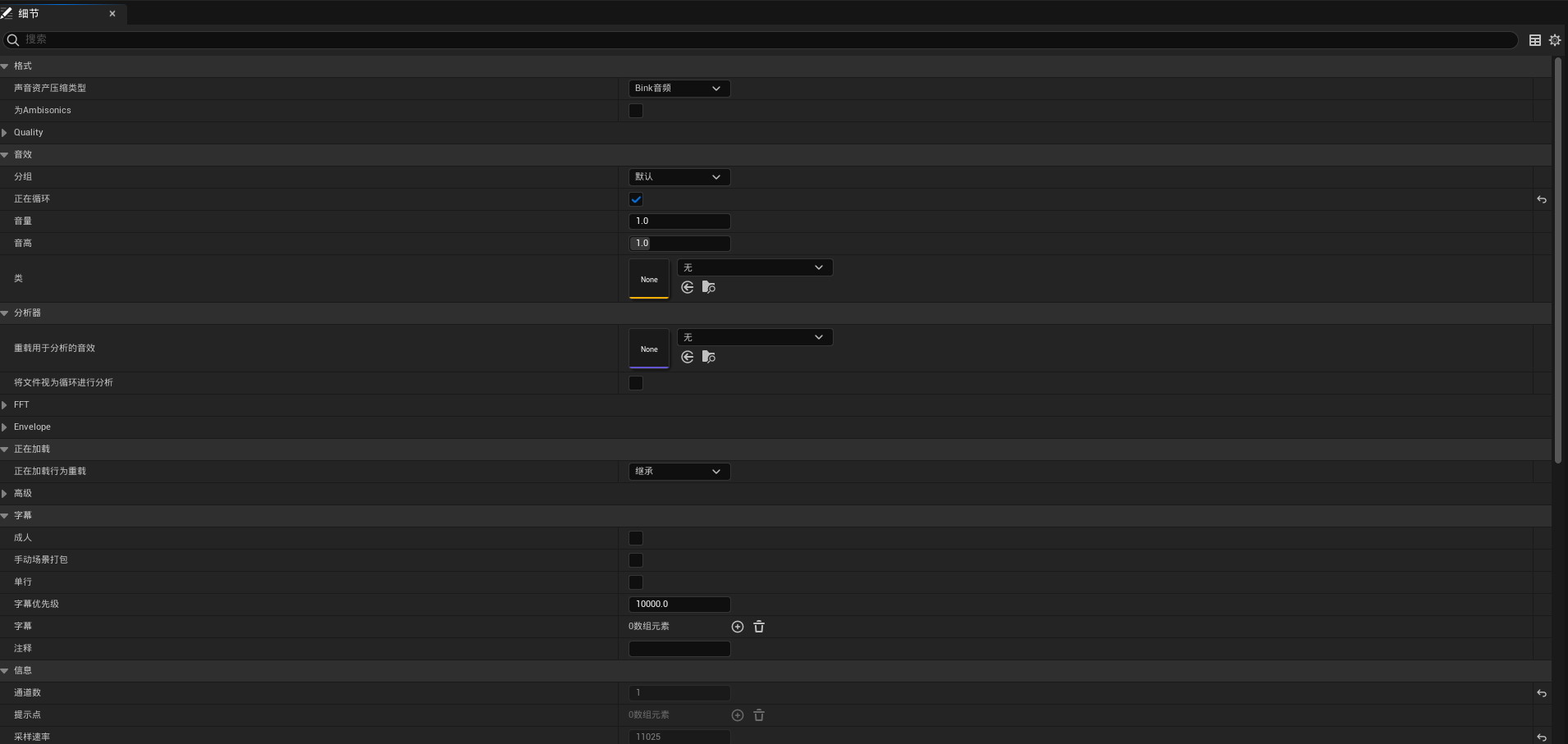
将这个Set Volume Multiplier节点连接在Set Material为Green和Move Component To节点中间



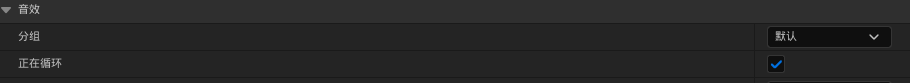
即完成了声音的绑定

接下来设置Hum音频组件则循环播放

在组件中点击Hum组件，找到它的音效，双击音效图标，进入音效的细节界面



在“音效”中找到“正在循环”，打勾



然后记得保存

即完成了平台音效的设置