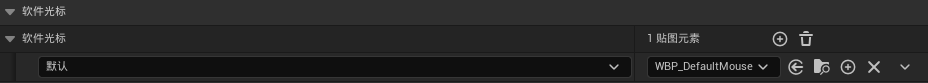
首先创建三张图片，作为鼠标光标的鼠标样式

创建WBP\_DefaultMouse，画布面板下加一个通用图像，将一个图片作为默认光标，设置“外观--笔刷--图像”

注意奥，WBP是可以设置动画的。

进入项目设置，找到“引擎--用户界面--硬件光标&软件光标”

设置一下软件光标



此时进入游戏就会发现鼠标光表已经被修改了

此时也需要关注一下PlayerController里的默认鼠标光标



注意，鼠标点击的默认位置是图像的中间点。所以需要对WBP进行适当调整。

然后实现光标的使用

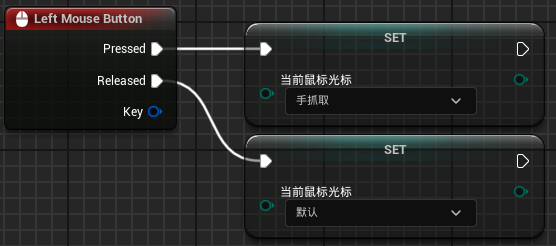
创建WBP\_ClickMouse

然后回到项目设置，再加一个默认光标



注意，手抓取这里可以选择除默认以外任何一个。因为是点击鼠标，逻辑功能类似于抓取，所以选择手抓取

然后进入PlayerController



节点的搜索名称为SetCurrentMouseCursor

## 选中目标高亮--材质

原理就是把当前的模型稍微扩大一点，然后把模型的正面变成透明，背面可渲染

新建材质命名为M\_OverlayLight

创建TwoSidedSign节点，连接到基础颜色。

既然用到TwoSidedSign节点，就应当勾选材质细节中的双面。此时将预览网格体设置为平面图元，就会发现一面白色一面黑色。

TwoSided节点输出的其实就是相当于把我们法线正面的地方都输出是一，负面的地方输出是零。

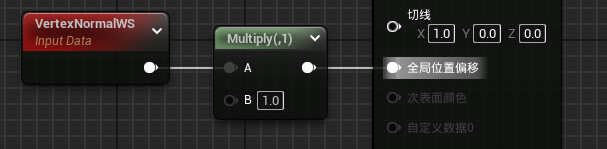
然后勾设置材质细节中的混合模式为已遮罩，将TwoSided节点赋值给不透明蒙版。但是我们要给TwoSided给一个1-x作为反转，这样让正面透明背面不透明。



然后再给材质做一个自发光就可以了

然后实现材质稍微大一点

创建VertexNormalWS（坐标），搞一下法线朝向。这个节点就相当于输出我们这个物体的表面的法线朝向。我们将他给到全局位置偏移。



最后，材质--着色类型设置为无光照。

进入场景，将这个材质赋值给“渲染--高级--覆层材质”

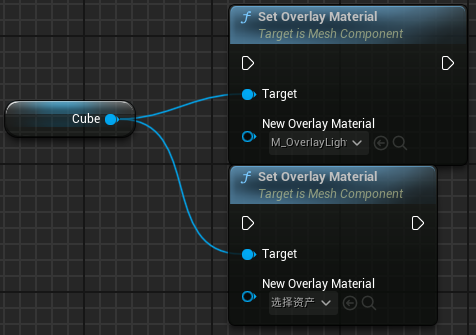
覆层材质：



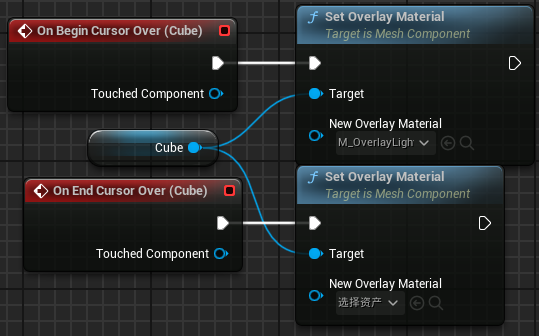
## 选中目标高亮--鼠标悬浮逻辑

在PlayerController中勾选“启用鼠标悬停事件”

创建Actor类蓝图命名为BP\_LightThing。我们将它作为一个父类，实现鼠标悬停发光。



然后在静态网格体（Cube）的事件中选择开始鼠标悬停时和结束鼠标悬停时



这两个事件节点还能返回鼠标悬停时碰撞到的组件。

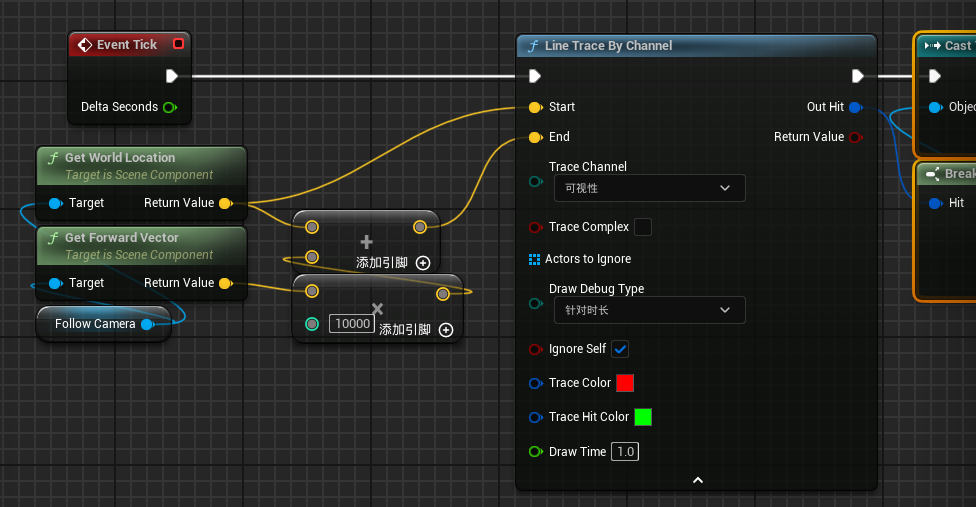
## FPS中心目标高亮

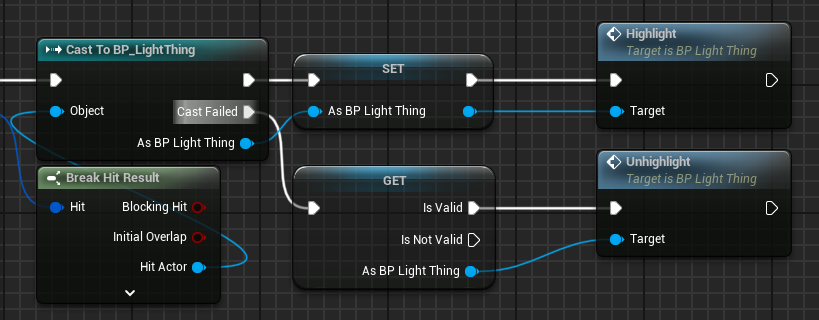
现在是FPS模式

使用射线追踪。

首先创建两个自定义事件封装暴露一下SetOverlayMaterial函数

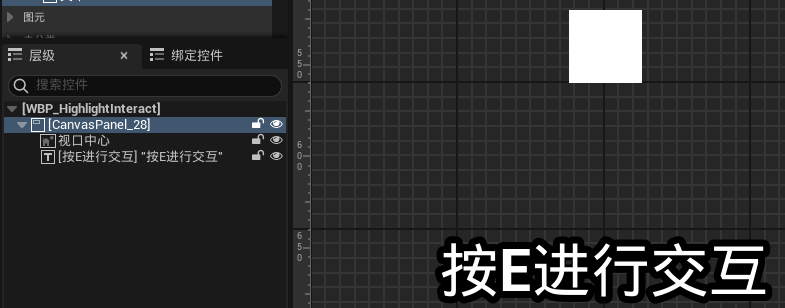
进入Character类



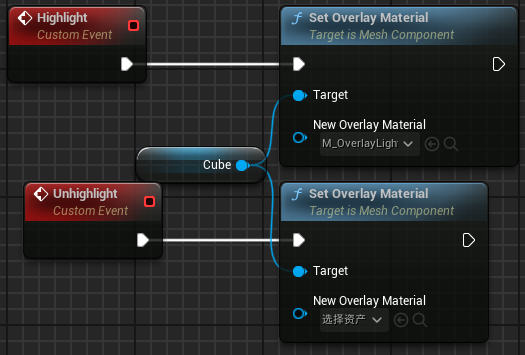


## 鼠标悬停选中返回为真

首先创建WBP\_HighlightInteract



在这后面CreateWidget即可



如果想不用在EventTick上多次CreateWidget，用个DoOnce节点即可。自己搞去吧

## 鼠标悬停选中进行交互

## Aura方法

Aura的假设是所有的敌人有相同的父类，在父类中使用蓝图接口实现。

这样，当我们的鼠标悬停在某Actor上时，会检查它是否实现了这个接口，如果是的话，就调用接口上的函数。

创建BPI\_HighlightInterface

创建函数Highlight和UnHighlight

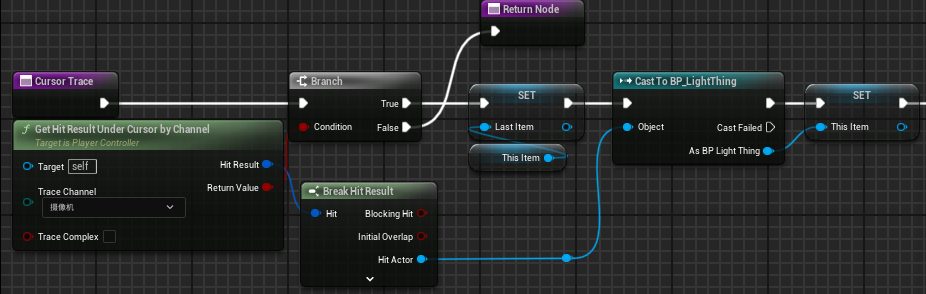
进入BP\_LightThing，接入接口。

我们可以先不关注函数的具体实现。我们先关注PlayerController，看究竟是哪个Actor在鼠标光标下，然后这个Actor如果实现了接口，我们可以调用Highlight。

我们需要获取鼠标光标下的任何Actor，这是每一帧都要进行的操作。

在PlayerController中，EventTick下进行相关操作

首先创建函数CursorTrace，并创建两个变量LastActor和ThisActor，变量都是BPI\_HighlightInterface类型。

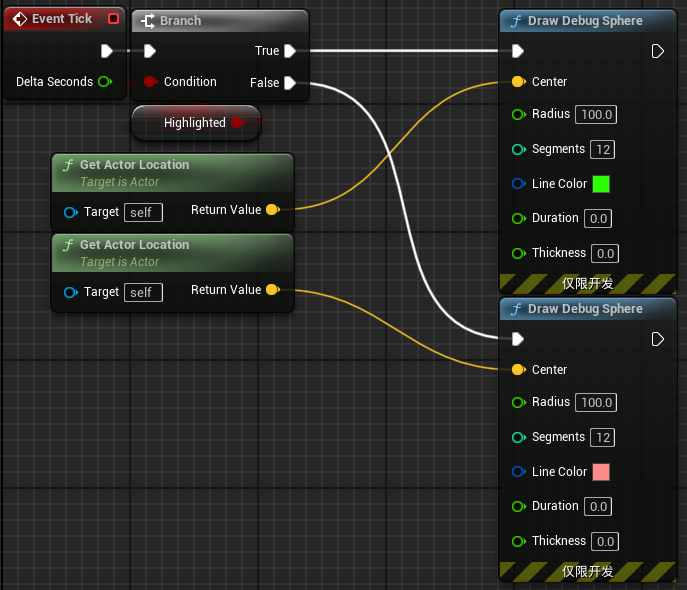




完成

然后进入BP\_LightThing，创建bool型变量Highlighted

首先在EventTick下测试



然后确保LightThing阻塞可见性通道

在碰撞预设中将Visibility设置为阻挡

最后在PlayerController的EventTick下调用CursorTrace函数，在BP\_LightThing下给蓝图接口函数设置Highlighted变量即可

现在进行测试，就可以发现正常运行

接下来实现高光后处理体积

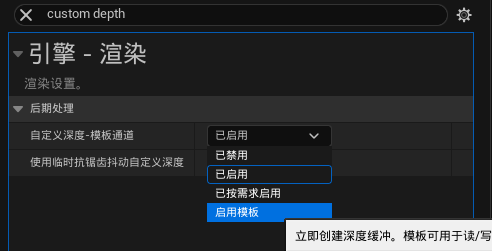
创建一个轮廓

在场景中创建一个后期处理体积，在细节中找到“后期处理体积设置--无限范围”，勾选

然后创建高光材质。这个材质在Aura资源的/Asset/Material中，名称是PP\_Highlight。该材质已经收录

在这个材质中，我们会用到自定义深度模板

进入项目设置，我们需要将custom depth改成启用模板（这一步很重要）



点击场景中的后期处理材质，在细节中找到“渲染功能RenderingFeatures--后期处理材质”



加上一个索引。

我们需要给他添加高光材质。

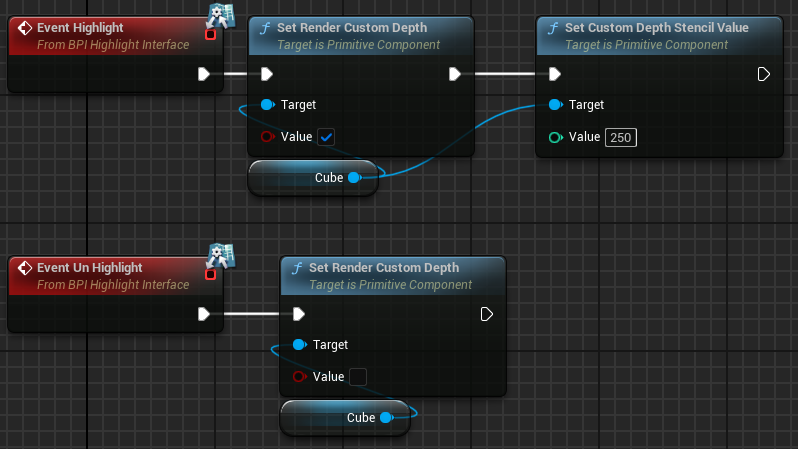
此时我们可以通过如下手段在引擎中看到轮廓：

点击场景中任何一个BP\_LightThing，在静态网格体细节中找到“渲染--高级--渲染自定义深度通道”，勾选后修改“自定义深度模板值”就能看到效果。

如果想改轮廓的粗细，进入PP\_Highlight修改最前面那个数值节点即可。

接下来要做的就是在游戏中看到轮廓。方法就是将“渲染自定义深度通道”变量设置为true并设置“自定义深度模板值”即可

我们要做的就是在BP\_LightThing的接口函数上实现这些功能



完成。

M\_Highlight收录版本

给粗细和颜色提升为参数，可以在创建材质实例的时候对颜色和参数进行设置。