第 9 课 蓝图节点(8)

10、蓝图（Blueprint）通信

有 4 中最常见的蓝图通信方式，分别为直接蓝图通信、事件调度器、蓝图接口和蓝图投射。

（1）蓝图投射

“类型转换为 ThirdPersonCharacter”节点

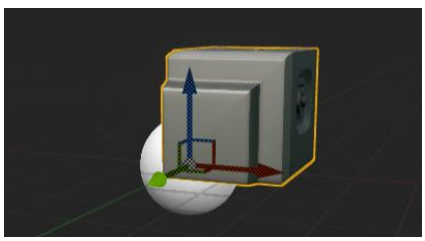
（2）直接蓝图通信

定义可编辑的变量，在蓝图实例对象上存储另一个蓝图的引用。

（3）事件调度器

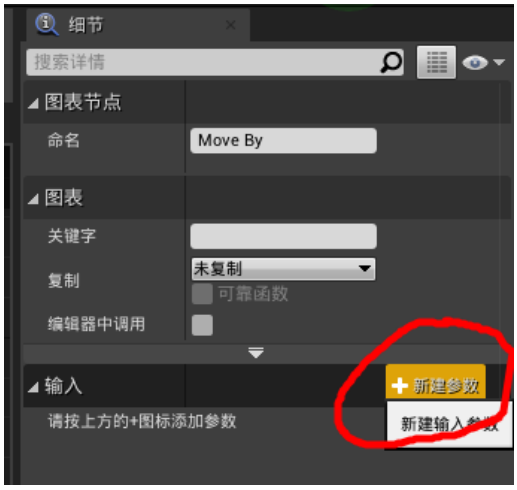
事件调度器（Event Dispatcher）适用于告知其他“正在监听的”蓝图：事件已发生。通过事件调度器，当事件发生时，正在监听的蓝图会作出反应，并相互独立地执行预设的一系列操作。

创建一个以 Actor 为父类的蓝图，命名为 MoveBox。打开 MoveBox 蓝图，添加一个静态网格体组件（Static Mesh），设置细节面板的静态网格体为 SM_CornerFrame。



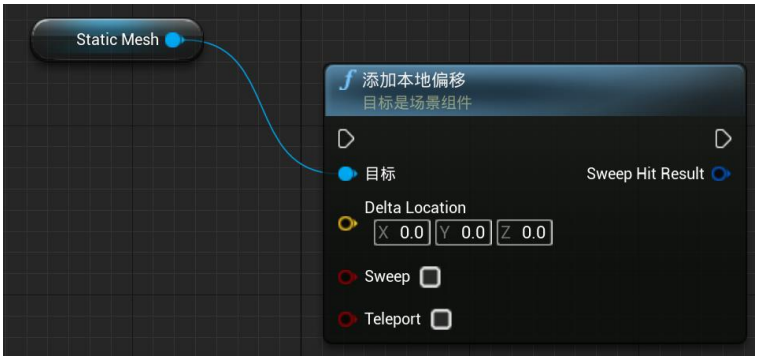
本节要实现通过键盘操作使 MoveBox 位置上升，当角色碰到 MoveBox 时会打印“Hello”。打开 MoveBox 蓝图，切换到事件图表编辑器，添加一个“自定义事件（Custom Event）”节点，命名为 Move By。

在细节面板为该节点添加一个浮点类型参数，命名为 Duration。

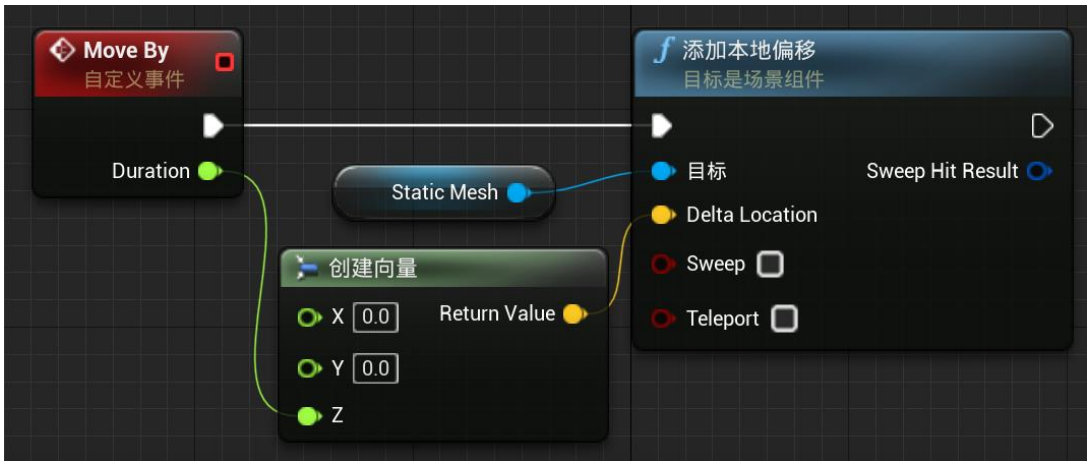


Move By 事件节点中多了一个引脚，当该节点被调用时，Duration 引脚将会作为 Move By 节点的输入引脚。

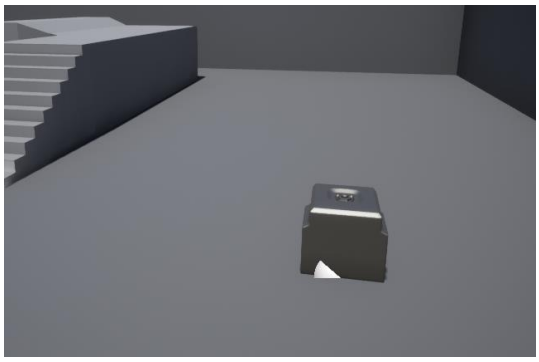
在组件面板中选中 StaticMesh，然后在事件图表中添加“添加本地偏移（StaticMesh）”节点，英文名称是“Add Local Offset (Static Mesh)”。该节点用于将指定目标的 Location 坐标增加 Delta Location。



添加“创建向量（Make Vector）”节点，连接引脚。

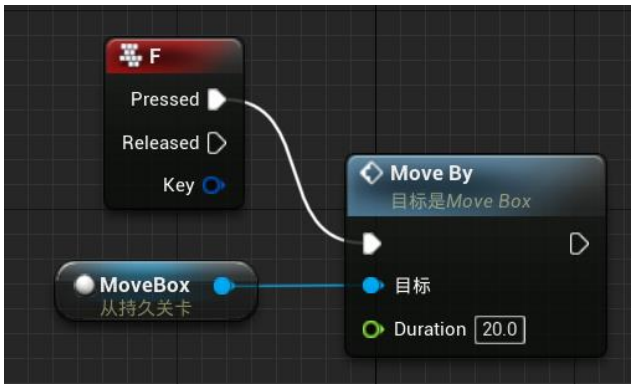


每执行一次 Move By 事件，Static Mesh 就会沿着 Z 轴方向增加 Duration 值。把 MoveBox 实例对象添加到游戏场景中，默认命名为 MoveBox。

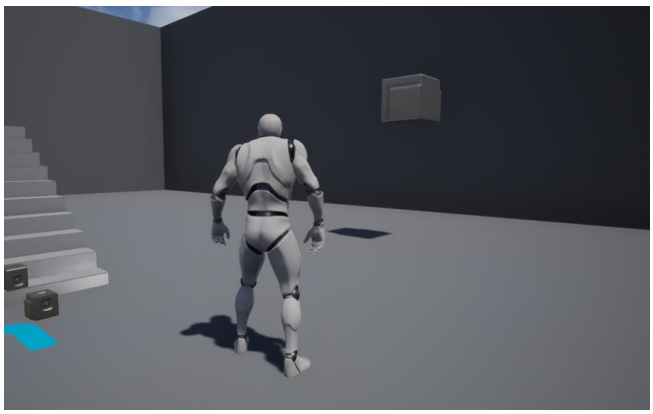


世界大纲视图中选中 MoveBox，打开关卡蓝图，在事件图表编辑器中按鼠标右键，添加“在 Move Box 2 上调用函数”（此处的数字 2 只是系统为 Move Box 设定的名字）>>“Move Box”>>“Move By”节点。

把 Move By 节点的 Duration 值修改为 20，并添加按键事件 F。



测试游戏，当按下 F 键后，MoveBox 沿 Z 轴上升 20 个像素单位。



这里我们实现了通过键盘控制 MoveBox 上升的机制，接下来要实现接触 MoveBox 后打印“Hello”的效果。

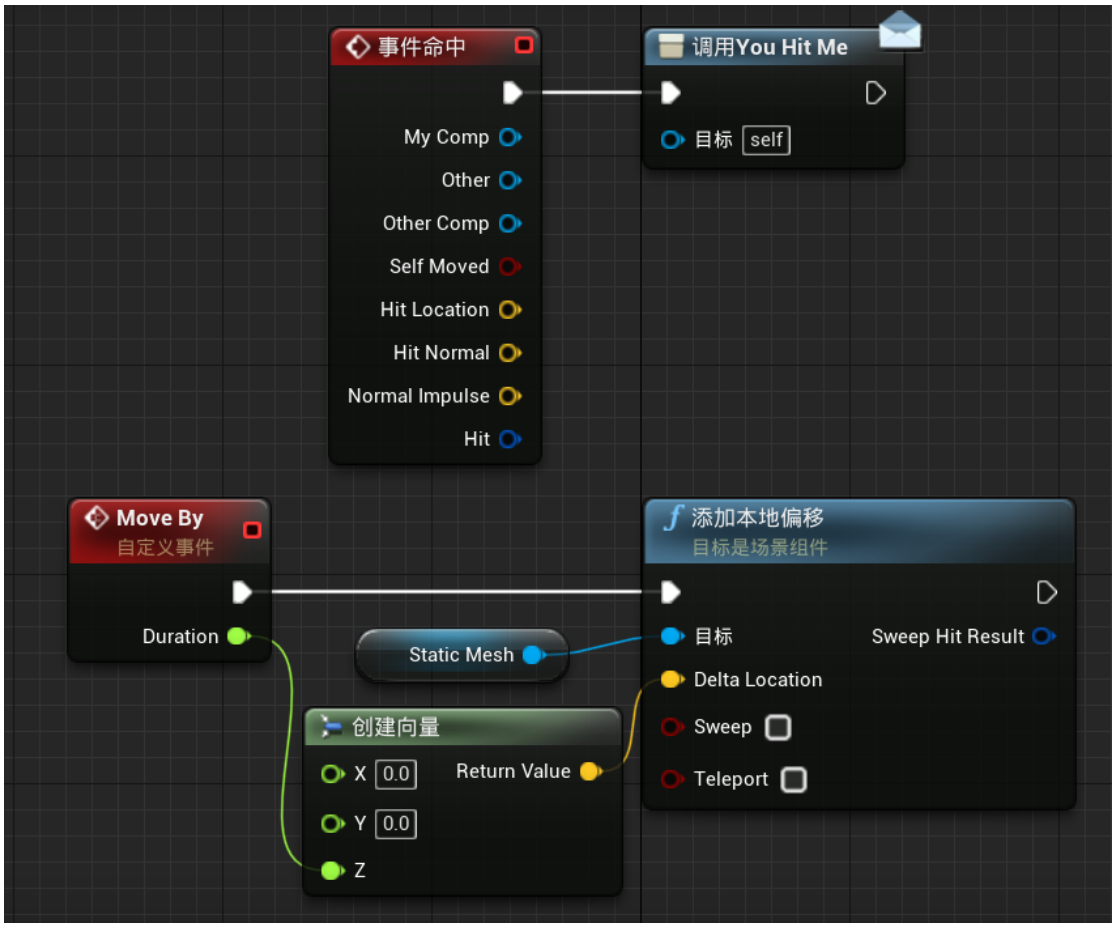
打开 MoveBox 蓝图，在“我的蓝图”面板里新增一个事件分发器，重命名为 YouHitMe。

在细节面板中可以设置它的类别，默认情况下的类别是“默认”；此外，还可以给它添加输入，从而允许向绑定到 YouHitMe 事件调度器的每个事件发送变量，使得数据不仅可以在类蓝图之间流动，还可以在类蓝图和关卡蓝图之间流动。

在 MoveBox 的事件图表编辑器中添加“事件命中（Event Hit）”节点，在当前蓝图的实例对象与其他 Actor 之间存在碰撞，且只要其中一个相关 Actor 的碰撞设置中把“模拟生成命中事件（Simulation Generates Hit Events）”设置为 true，“事件命中”节点就会执行，它不同于“事件 Actor 开始重叠（Event Actor Begin Overlap）”和“事件 Actor 结束重叠（Event Actor End Overlap）”节点只在两个 Actor 重叠状态变化的一瞬间执行一次，“事件命中”节点是在两个 Actor 之间存在碰撞的时间段内不断地被调用执行。

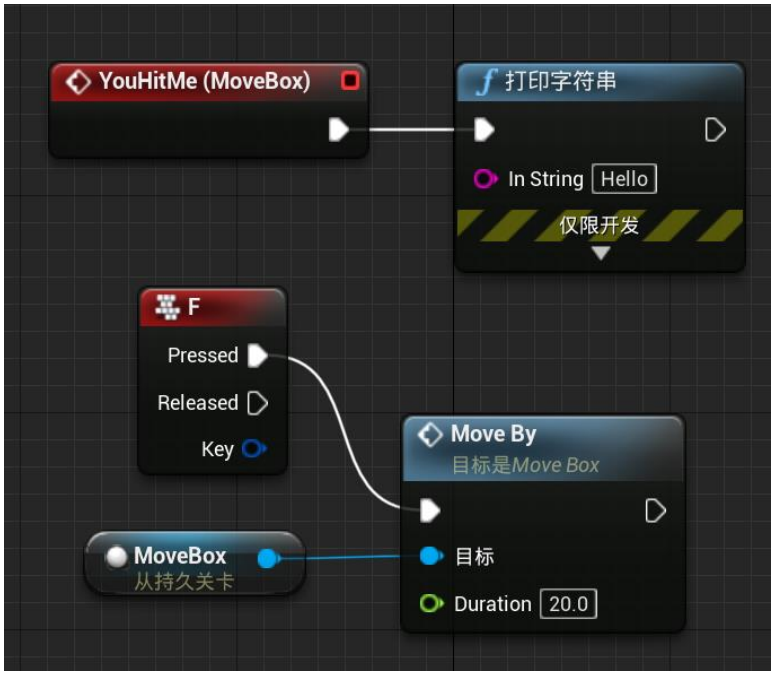
拖动“事件命中”节点的执行引脚至空白处松开，找到“调用(Call) You Hit Me”节点。

目前在 MoveBox 蓝图里，所有节点连接如下：



相对于 Move By 自定义事件相当于在 MoveBox 类蓝图里定义了一个成员函数，供其他蓝图调用；You Hit Me 事件分发器是主动告知其他蓝图，发生了事件命中，即有物体（Actor）撞到自己了。

世界大纲视图中选中 MoveBox，打开关卡蓝图，添加“You Hit Me”节点，连接后续的打印字符串节点。关卡蓝图的节点连接（包括之前关于调用 Move By 事件的连接）如下：



运行游戏，当玩家尝试碰撞 MoveBox 对象时，屏幕会一直输出 Hello。

可以从事件分发器和自定义事件的对比中看到，适合作为事件分发的，是那些当事者无法预期何时发生的事件，例如当前对象何时会被撞到；适合定义为自定义事件供人调用的，是那些固定的明确的功能，例如当按下某个按键时会触发移动或者输出。

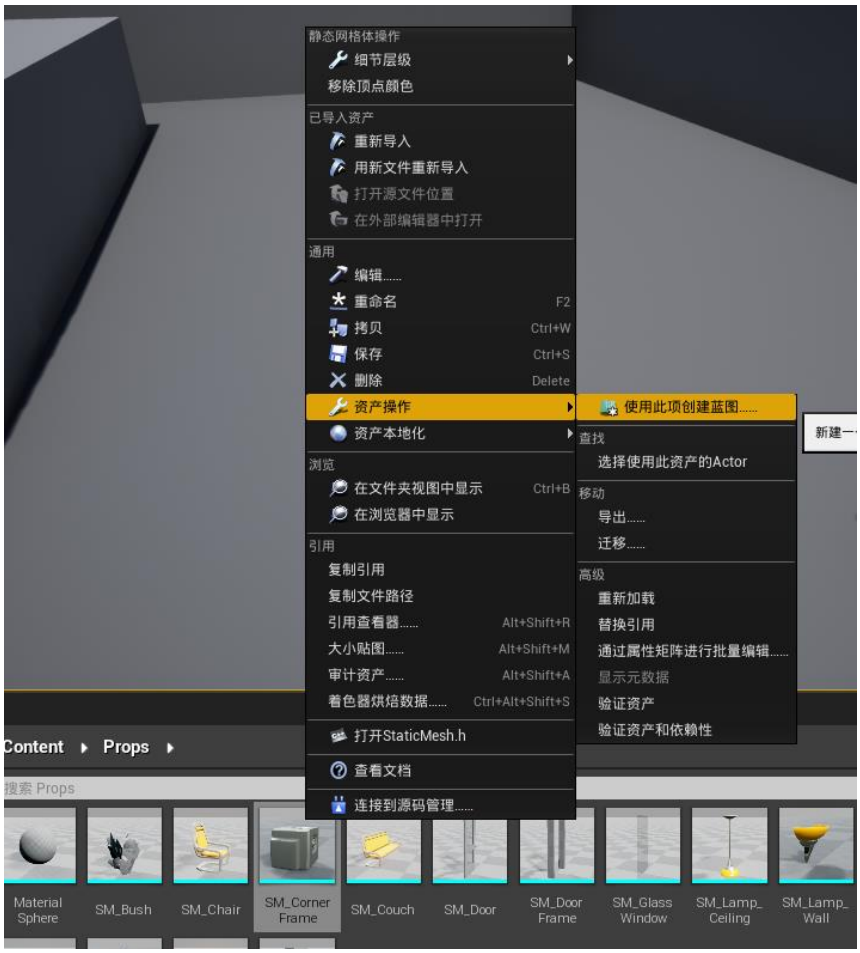
（4）蓝图接口

蓝图接口（Blueprint Interface）可以实现多种类型对象互动。适合使用蓝图接口的应用场景是：在数个蓝图中存在一些相似的功能，但在调用后执行不同的效果。例如猫类和狗类都有一个输出叫声函数，因此可以创建一个把两者视为相同对象的蓝图接口，在任意一个对象需要发出叫声时调用自己的叫声函数，使得发出不同的声音。本节中将利用“由通道检测线条（LineTraceByChannel）”节点检测对象类型，针对不同类型的对象通过蓝图接口实现不同的移动效果。

在学习接口之前，先了解“由通道检测线条”节点。该节点可以在规定的 Start 和 End 向量坐标区间绘制射线，通过射线检测碰撞物，返回是否发生碰撞的布尔数据（Return Value）和碰撞数据（Out Hit）。

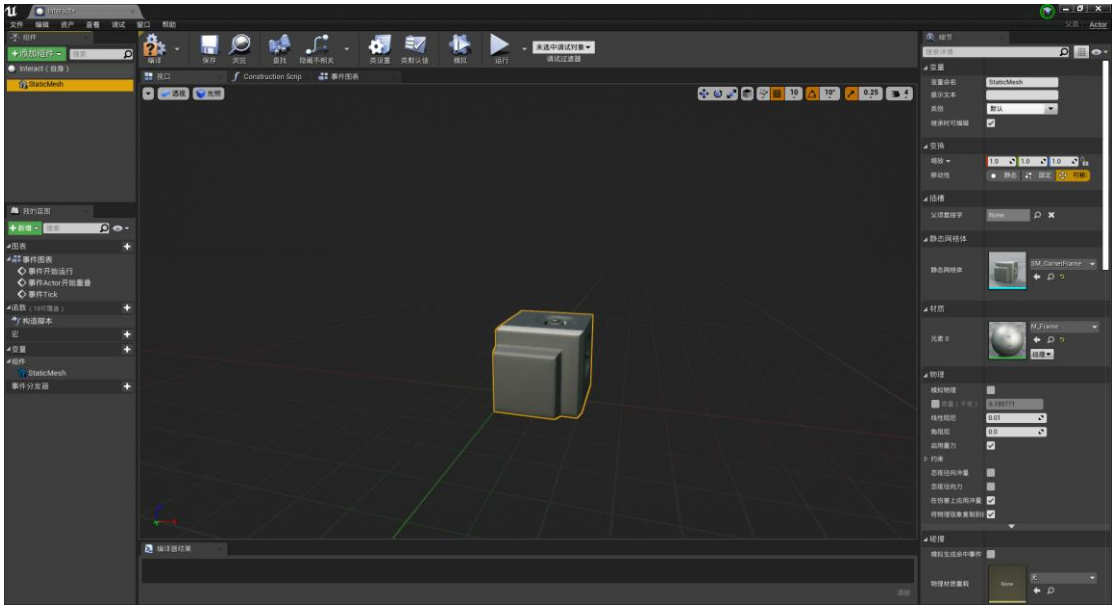
之前已经学过了好几种创建蓝图后向其中添加组件的方法，这次再学一种新的方法在创建蓝图的同时完成添加组件的工作。

在内容浏览器，找到待添加的静态模型资源 StartContent>>Props>>SM_CornerFrame，右键点击，选择“资产管理”>>“使用此项创建蓝图”。

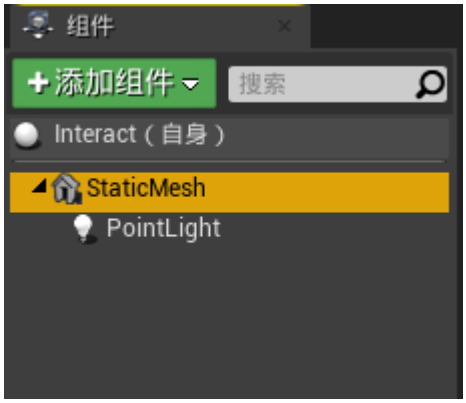


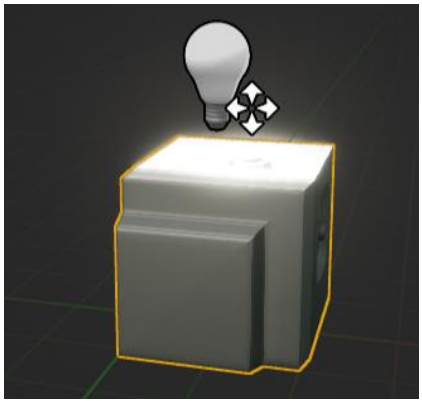
在弹出的窗口中选择蓝图存储的文件夹 ThirdPersonBP>>Blueprints，并修改蓝图命名为 Interact。

Interact 蓝图会自动打开，界面与其他方式建立的蓝图基本相同。区别在于 SM_CornerFrame 组件默认作为根组件存在于 Interact 蓝图中（其他方法建的蓝图，默认根组件是个球），默认命名为 StaticMesh。



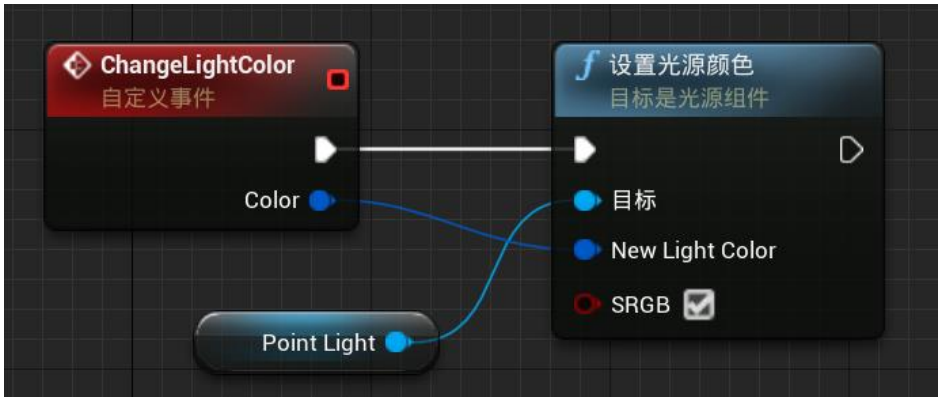
再添加一个点光源组件 PointLight。





打开 Interact 蓝图的事件图表编辑器面板，添加一个自定义事件（Custom Event）节点。为事件节点添加一个线性颜色（Linear Color）类型的输入参数 Color。

在组件面板把 PointLight 拖入蓝图中，通过该节点找到“设置光源颜色（Set Light Color）”节点。

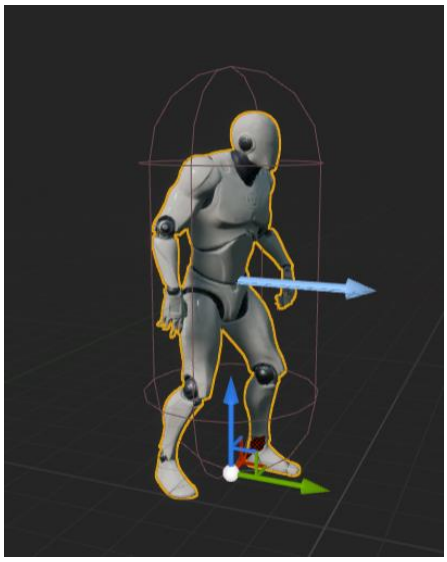


编译并保存 Interact 蓝图。返回主界面的内容浏览器，打开 ThirdPersonBP>>Blueprints>>ThirdPersonCharacter 蓝图，切换到事件图表编辑器面板，把原先已经添加过的“鼠标左键”事件节点断开后续的连接（如果没有就添加节点），再添加一个“由通道检测线条（LineTraceByChannel）”节点。

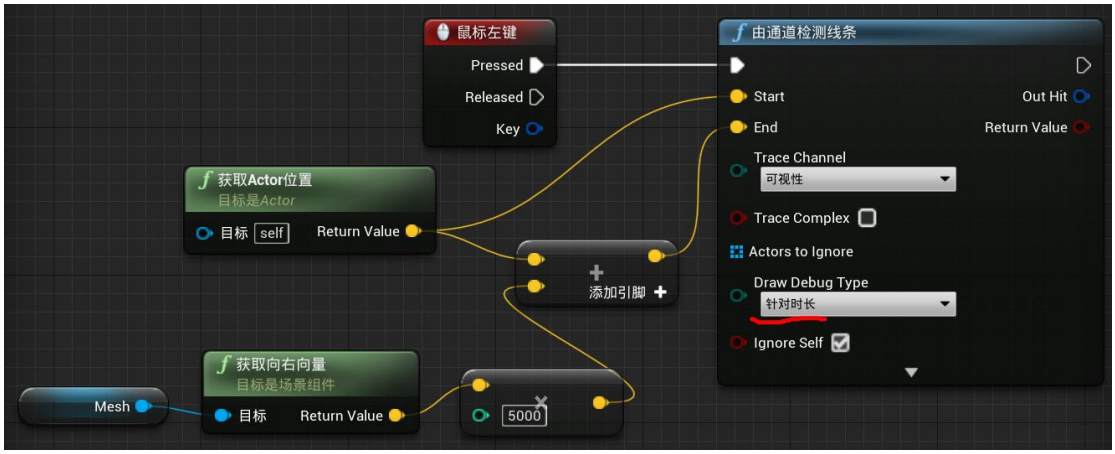
因为要利用“由通道检测线条”节点检测角色前方是否有物体存在，所以节点的起始坐标应设置为角色的坐标。添加“获取 Actor 位置（Get Actor Location）”节点。

终点和角色要有一定距离，且起点与终点的连线方向应为角色正面的法线。添加“获取向右向量(Mesh)（Get Right Vector(Mesh)）”节点。

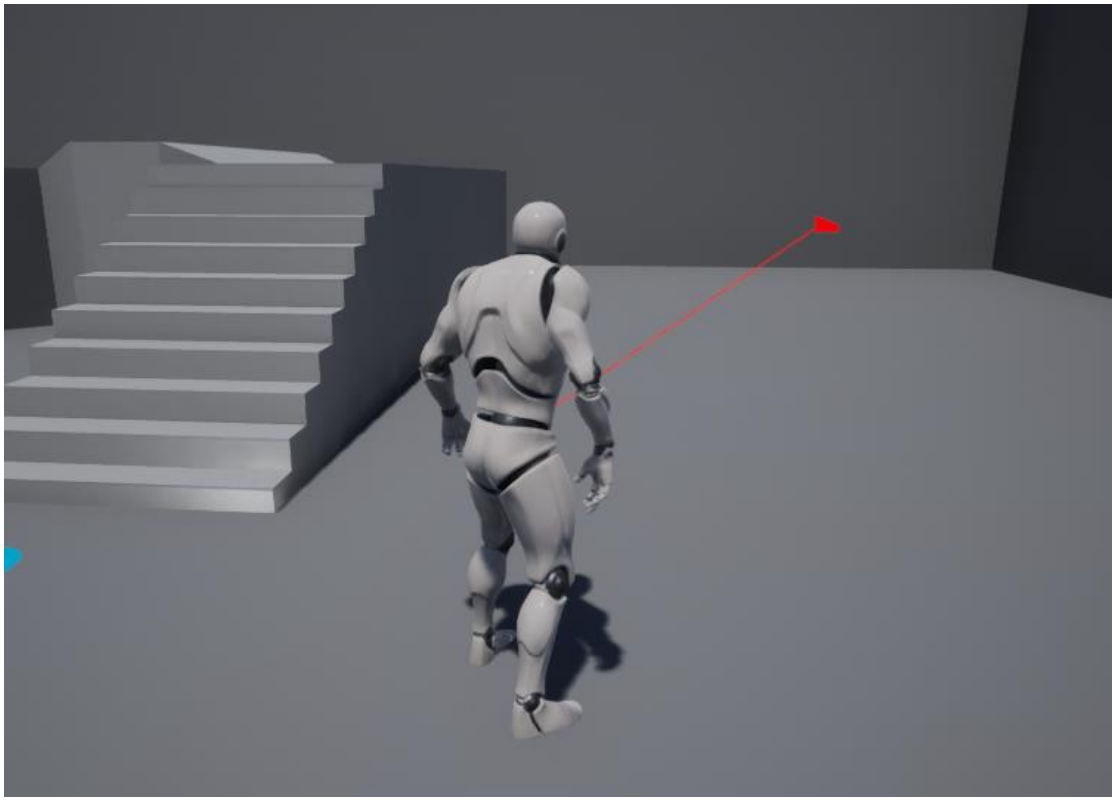
这里不是获取向前向量(Mesh)的原因是，ThirdPersonCharacter 蓝图中，角色的模型 Mesh 脸对的方向（淡蓝色长箭头）不是向前向量（红色箭头方向），而是向右向量（绿色箭头方向）。



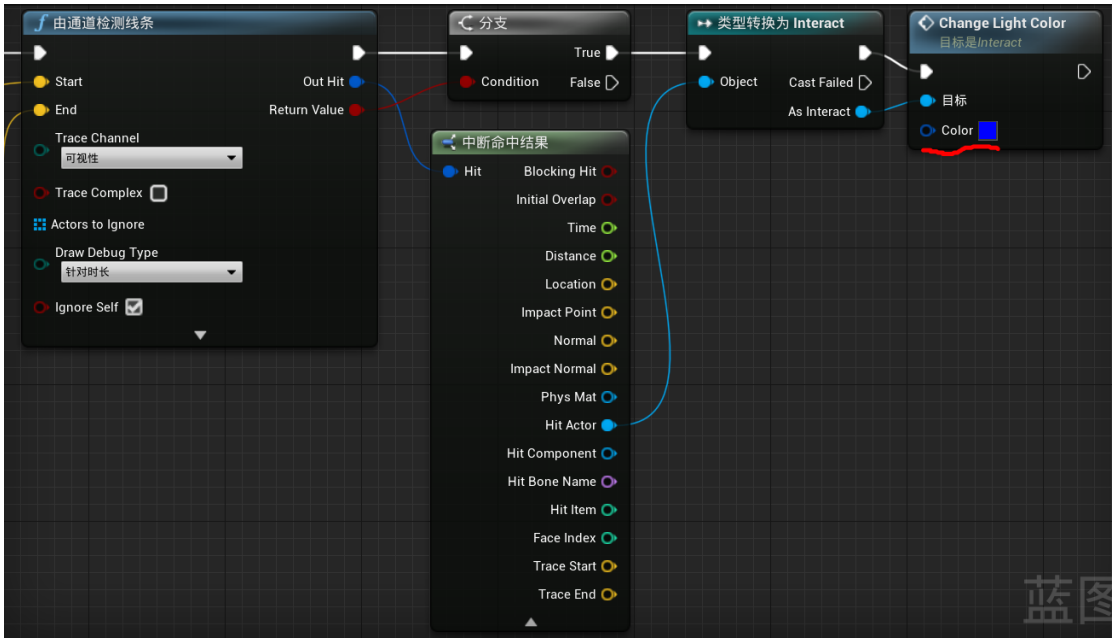
再添加“向量×整数”节点和“向量+向量”节点，连接连线。把“由通道检测线条”节点的 Draw Debug Type 引脚的值改为“针对时长”。



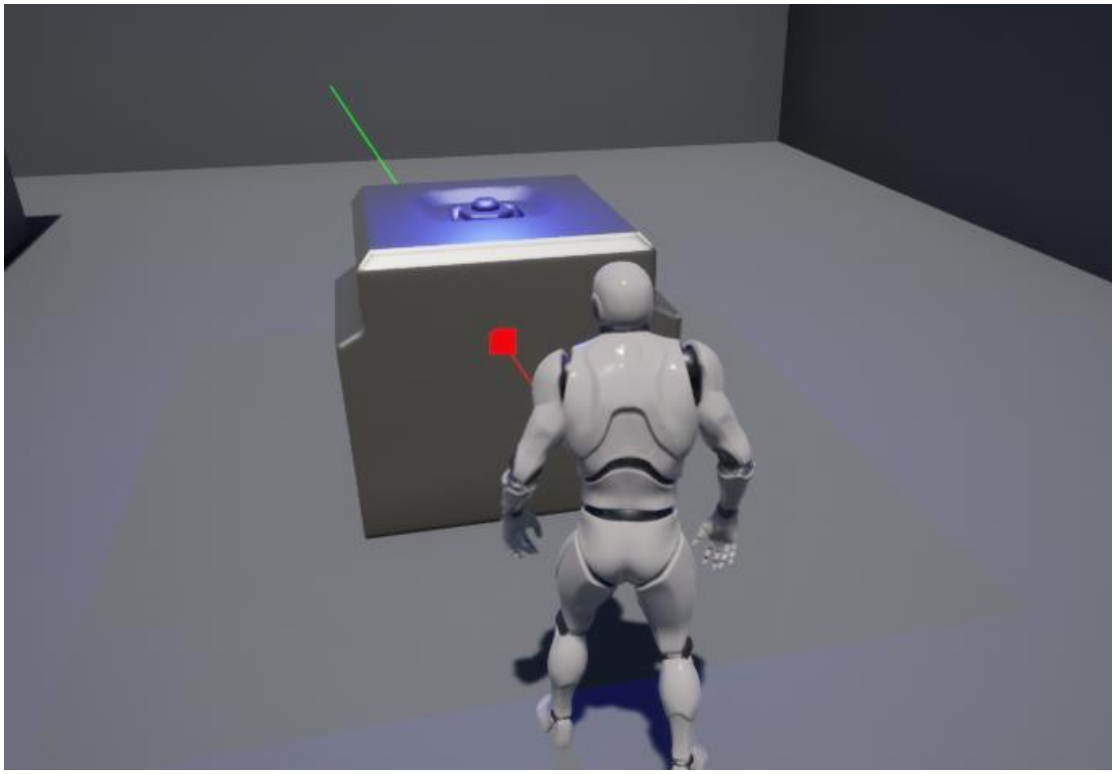
编译保存 ThirdPersonCharacter 蓝图，运行游戏。当按下鼠标左键时，会看到角色正前方有一条红色的射线出现，射线过一阵子会消失。



接下来要实现：当射线检测到 Interact 实例对象后，Interact 对象中的 PointLight 组件的灯光颜色会改变。返回 ThirdPersonCharacter 蓝图，添加“分支（Branch 或 if）”节点，该节点根据检测范围内有无物体来控制执行不同的行为。添加一个“中断命中结果（Break Hit Result）”节点，该节点中文翻译特别不合适（“分解命中结果”会更合适），用于分解 Out Hit 结构体，获得其中的成员值。添加“类型转换为 Interact（Cast To Interact）”节点，尝试把检测到的对象投射成 Interact 类型对象。从“类型转换为 Interact”节点的“As Interact”引脚，找到已在 Interact 类中定义过的“Change Light Color”节点，修改 Color 引脚的颜色。

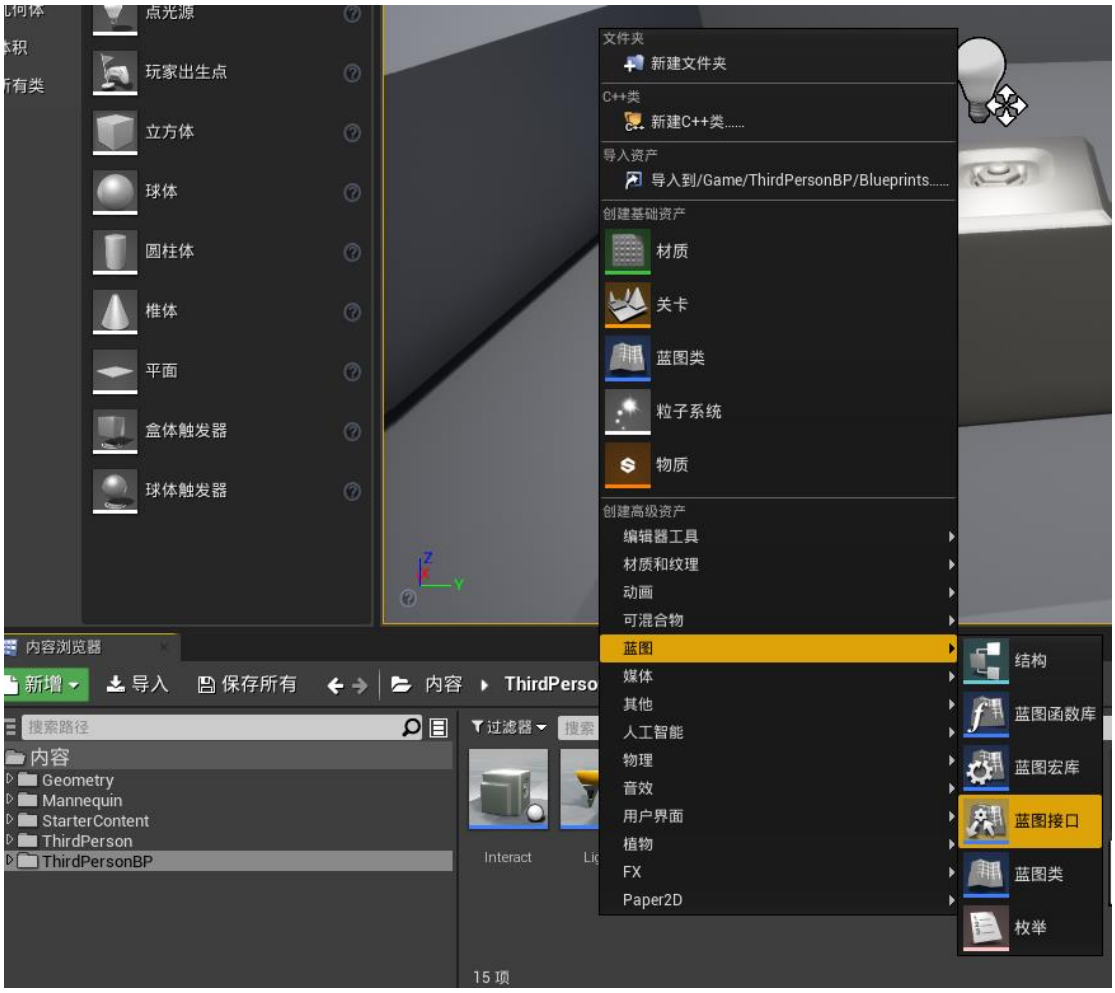


把 Interact 蓝图拖入游戏场景中创建一个实例对象，该对象中的 PointLight 组件默认呈现白色灯光。运行游戏，点击鼠标，当检测线条检测到 Interact 对象时，PointLight 的颜色发生变化。

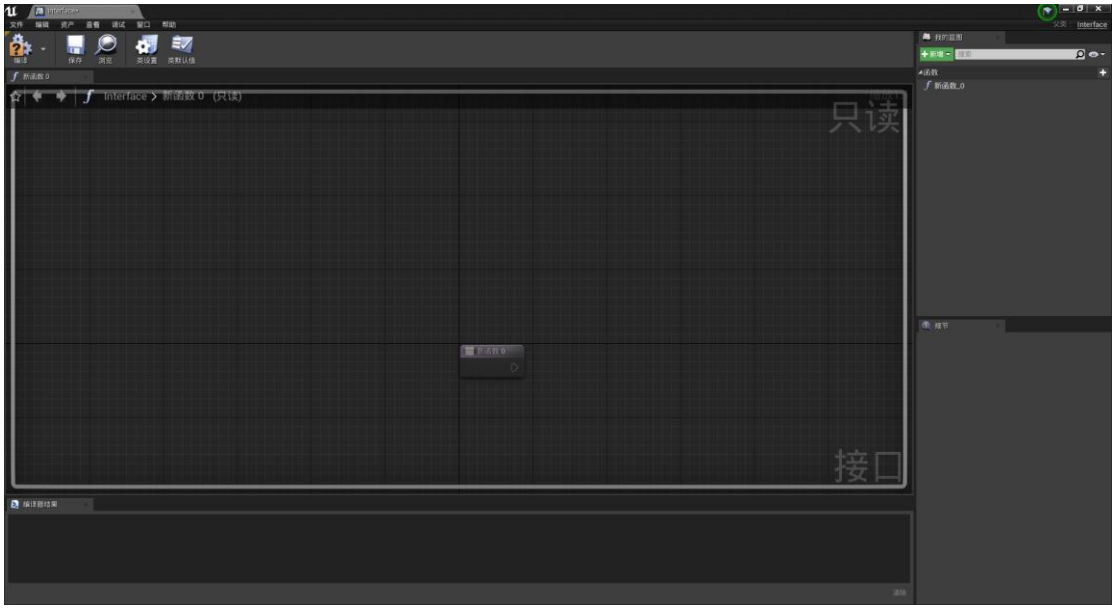


了解“由通道检测线条”节点及一系列相关节点的用法后，接下来将利用它们来学习蓝图接口的用法。

在主界面的内容浏览器面板中，鼠标右键，添加“蓝图>>蓝图接口”，命名为Interface。



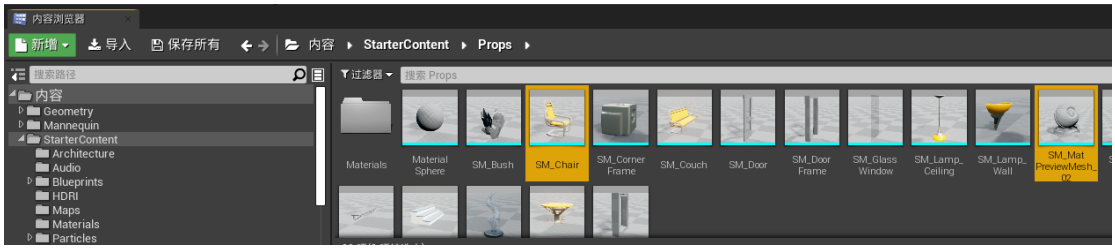
打开 Interface 蓝图接口。

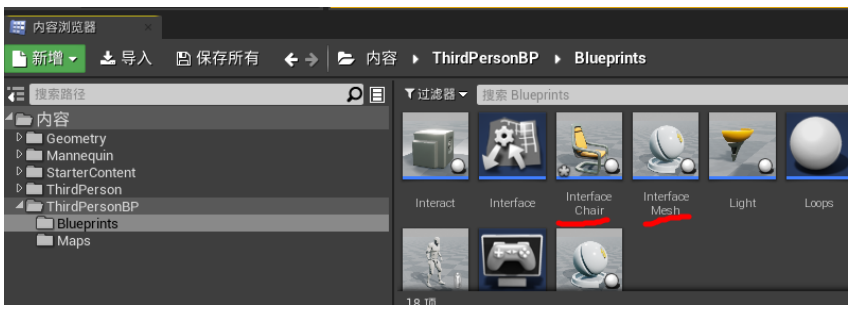


蓝图接口中目前只有“新函数 0（New Function 0）”节点，但该函数甚至整个图表编辑器面板的操作权限都为只读操作（Read-Only），因此接口只提供函数名，不可在蓝图接口中实现函数体。

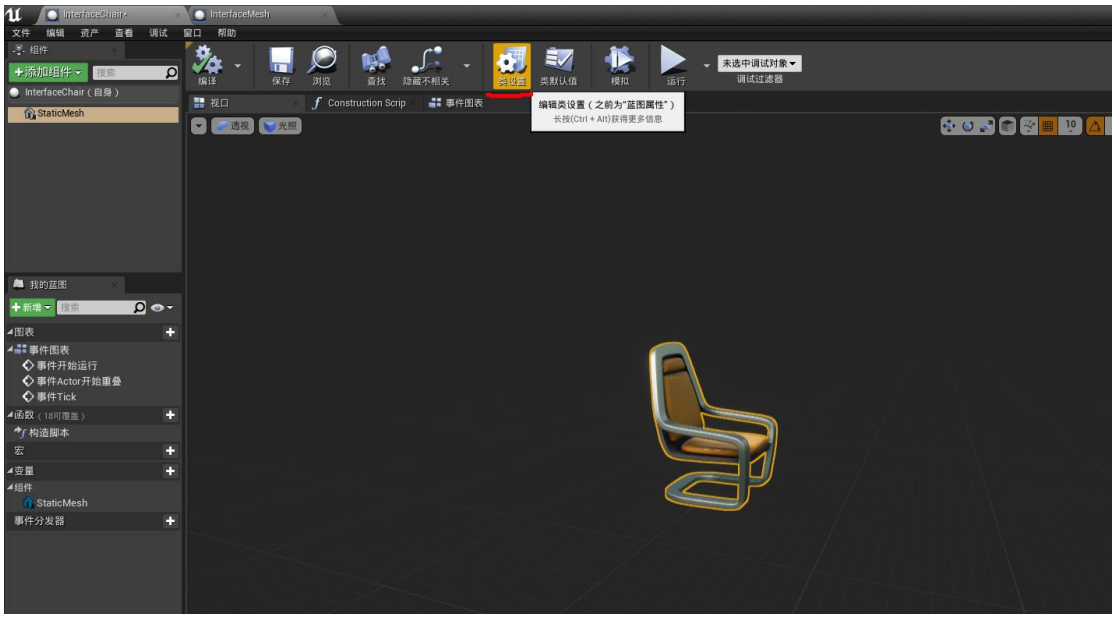
在细节面板把“新函数 0”节点重命名为 OnInteract，编译并保存蓝图接口。

在主界面的内容浏览器中，找到 SM_Chair 和 SM_MatPreviewMesh_02 静态模型，分别以它们为根组件创建两个蓝图（资产操作>>使用此项创建蓝图），各自命名为 InteractChair 和 InteractMesh。

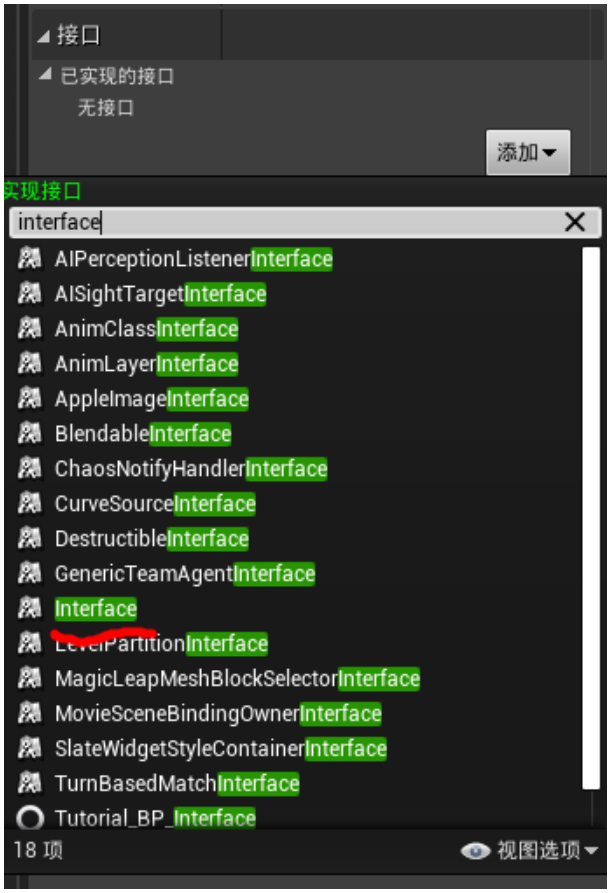




在 InterfaceChair 蓝图中，单击工具条中的“类设置（Class Setting）”。



在右侧的“细节面板>>接口>>已实现的接口”中，按下“添加”按钮，该按钮可以添加当前蓝图需要使用的蓝图接口。在列表中选择之前定义的蓝图接口 Interface。



可以看到，InterfaceChair 蓝图中，可执行的蓝图接口 Interface 被添加到已实现的接口类目下。

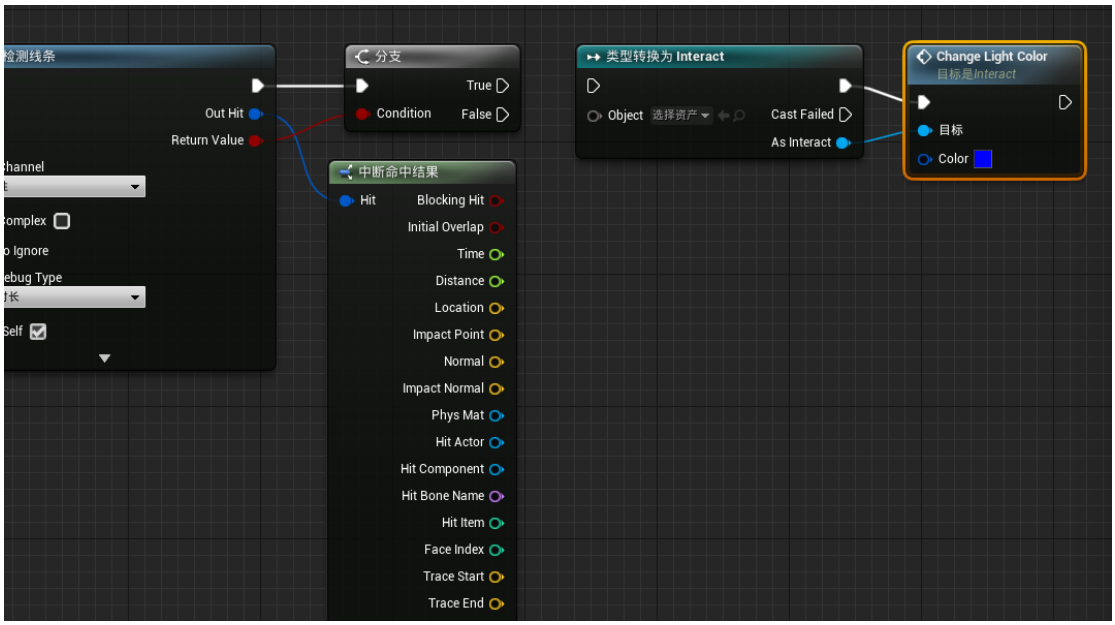
在 InterfaceChair 蓝图的事件图表面板中添加“事件(Event) On Interact”节点。

现在就可以实现 On Interact 函数，即将需要执行的节点与“事件 On Interact”节点通过执行引脚相连即可。添加“添加 Actor 本地偏移(Add Actor Local Offset)”节点，它可以控制 Actor 对象产生位移。

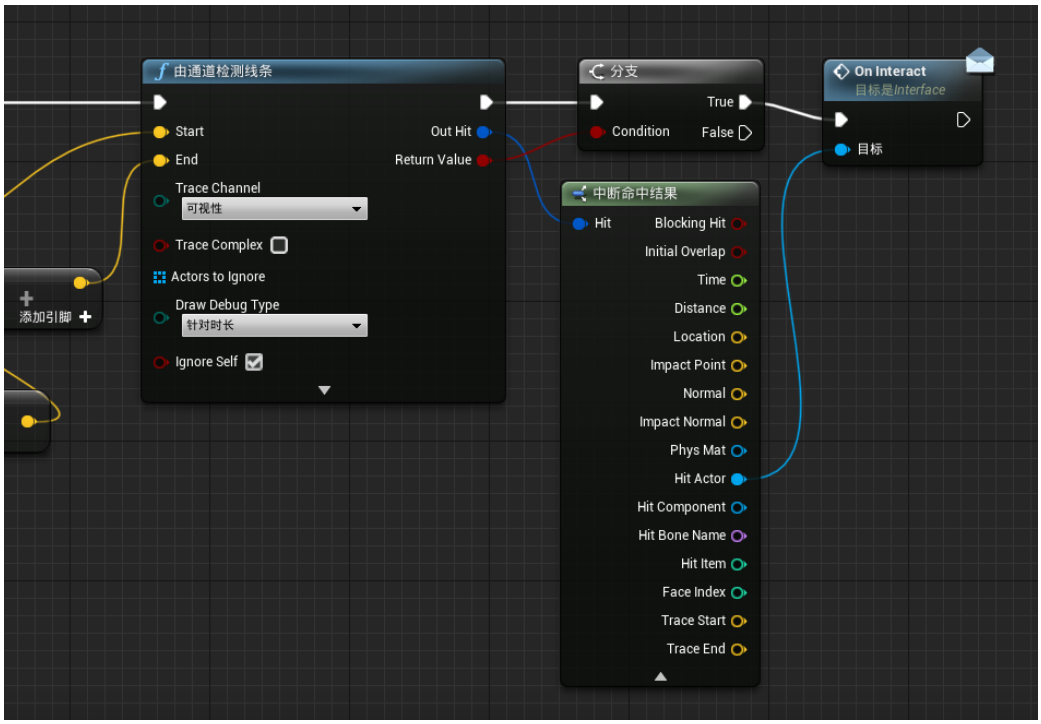
同样道理，在 InterfaceMesh 蓝图中的类设置里添加已实现的接口：Interface。

在事件图表面板中添加相同的蓝图节点，把“添加 Actor 本地偏移”节点中的 Delta Location 的 Y 设置为 10。编译保存蓝图。

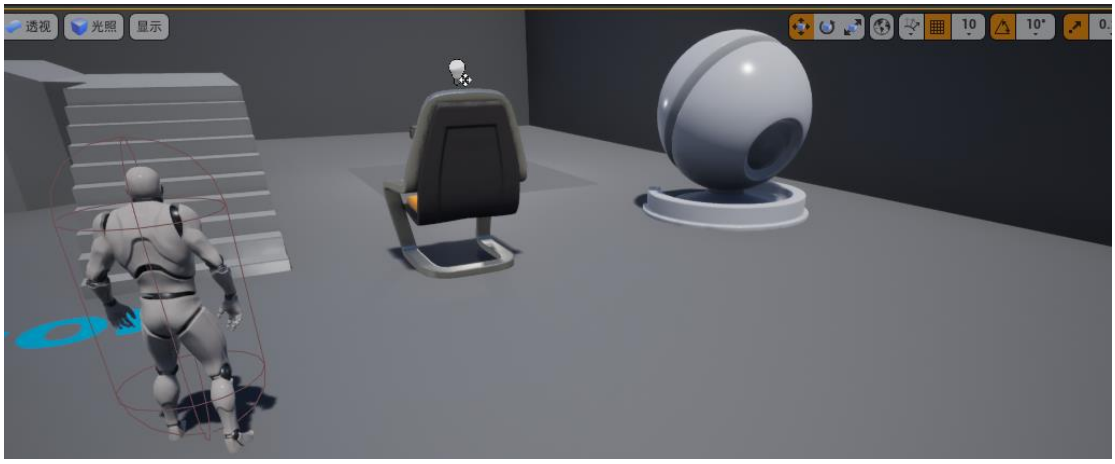
返回 ThirdPersonCharacter 蓝图，先断开“类型转换为 Interact”节点和“分支”节点、“中断命中结果”节点的连线。



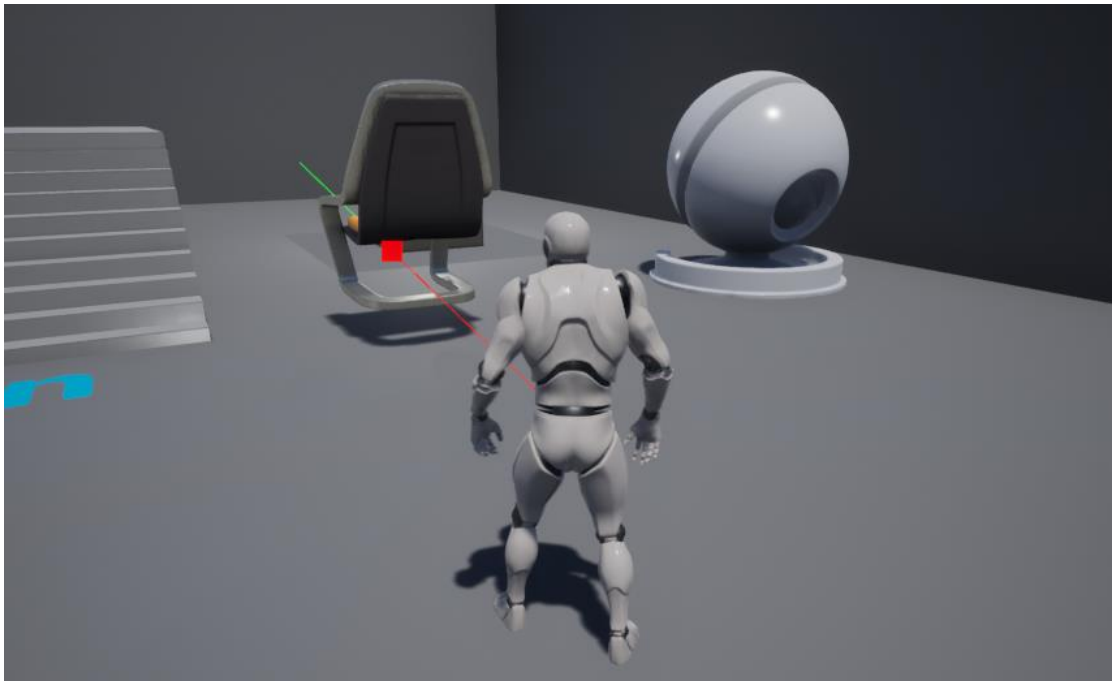
添加“On Interact”节点，该函数是 Interface 蓝图接口中定义的接口函数。当“由通道检测线条”节点在检测范围内检测到有物体存在时，则执行被命中的物体里实现的 On Interact 函数。如果命中物体没实现 Interface 中的 On Interact 函数，则什么事都不会发生。



在主界面游戏场景中，添加 InterfaceChair 和 InteractMesh 实例对象。



运行游戏，在游戏中按鼠标左键进行物体检测。每当检测到 InterfaceChair 实例对象，该对象沿 Z 轴上升 10 像素；每当检测到 InterfaceMesh 实例对象，该对象沿 Y 轴移动 10 像素。



总结一下跟蓝图通信有关的节点：“获取类的所有 Actor” 节点、自定义事件、“可编辑实例”变量、“类型转换为...”节点、事件调度器、蓝图接口。

作业：思考课程设计要实现的功能点，提交项目设计文档。