第 7 课 蓝图节点(6)

6、For 循环和 While 循环

本节介绍 4 种循环节点，分别是 ForLoop、ForLoopWithBreak、ForEachLoop 和 WhileLoop 节点。

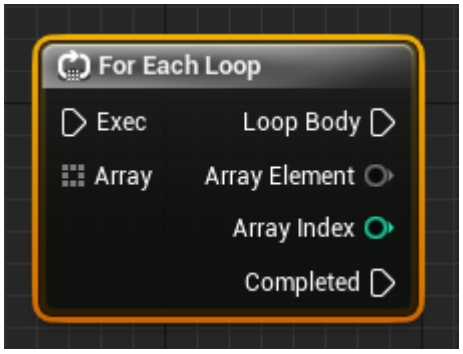
（1）ForLoop 节点

（2）ForLoopWithBreak 节点

（3）ForEachLoop 节点

ForEachLoop 是针对数组（Array）的循环节点。本节通过打印 String 数组变量，以及切换一组 Actor 的显示状态，来学习 ForEachLoop 节点的使用。

继续使用之前的 Loops 蓝图，切换到 Loops 蓝图的事件图表编辑器，添加一个 ForEachLoop 节点。

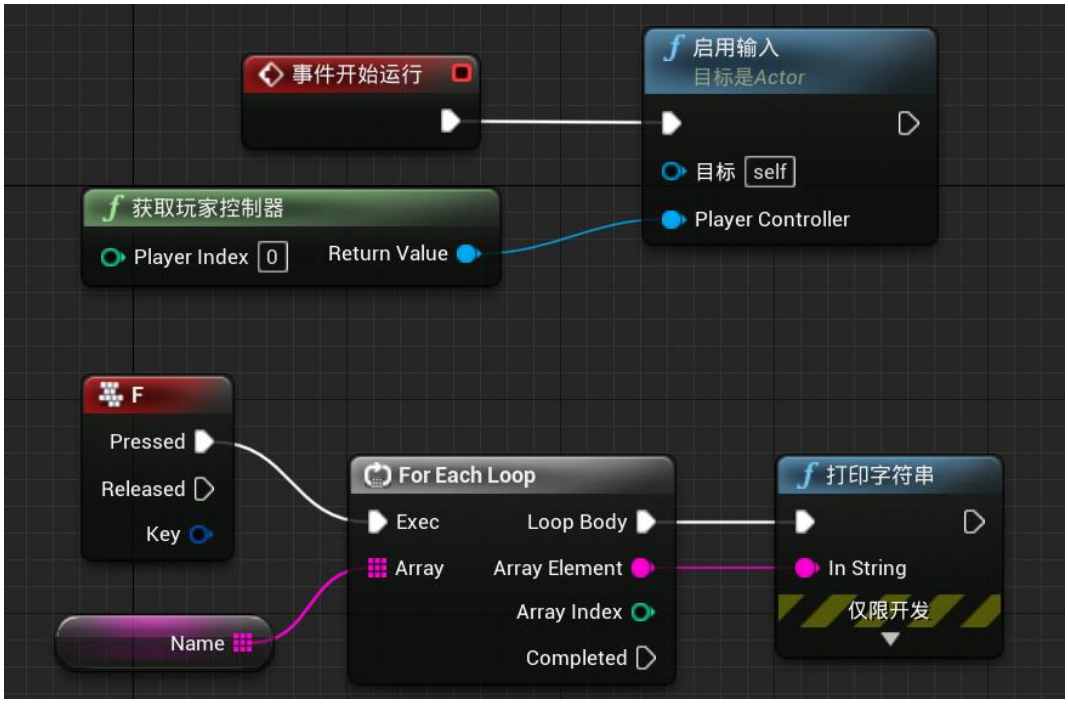


ForEachLoop 节点针对左侧 Array 数组变量，从数组的第一个元素开始执行 Loop Body，直到执行完数组最后一个元素。循环的次数等于 Array 的元素个数。参数 Array Element 和 Array Index 表示当前循环调用元素的值和下标。引脚 Array 和 Array Element 的颜色默认为灰色，它们会根据传递给 Array 参数的数组元素类型自动匹配类型。

在 Loops 蓝图中创建 String 类型的数组变量 Name，编译后在“默认值”处添

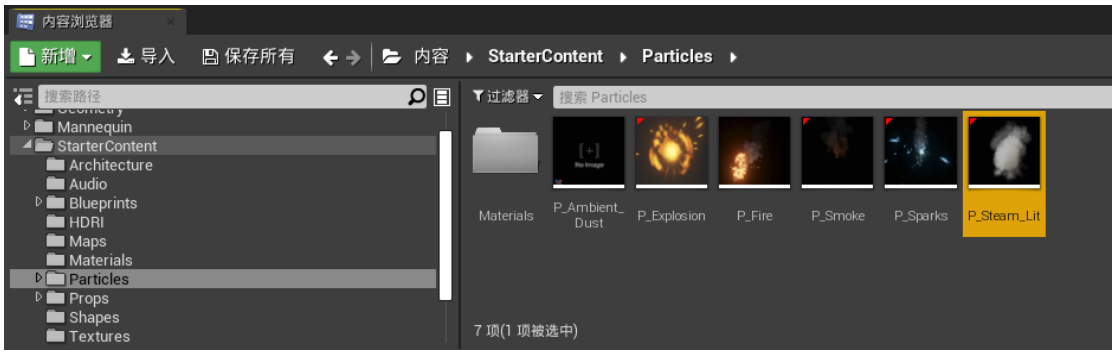
加 3 个元素：Alice、Bob 和 Tom。

把创建好的数组变量 Name 拖入事件图表编辑器中，删除上一节课为 ForLoopWithBreak 循环节点而创建的蓝图，改为把按键 F 节点连接入刚创建的 ForEachLoop 节点。

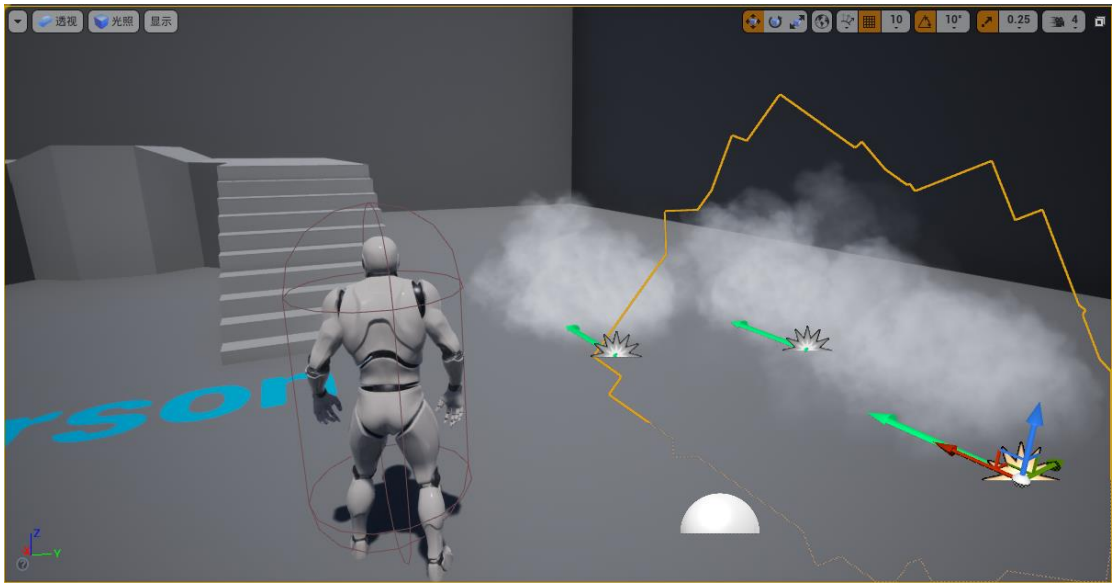


编译蓝图，运行游戏，当按下 F 键后，Name 数组中的 3 个元素依次被打印出来。

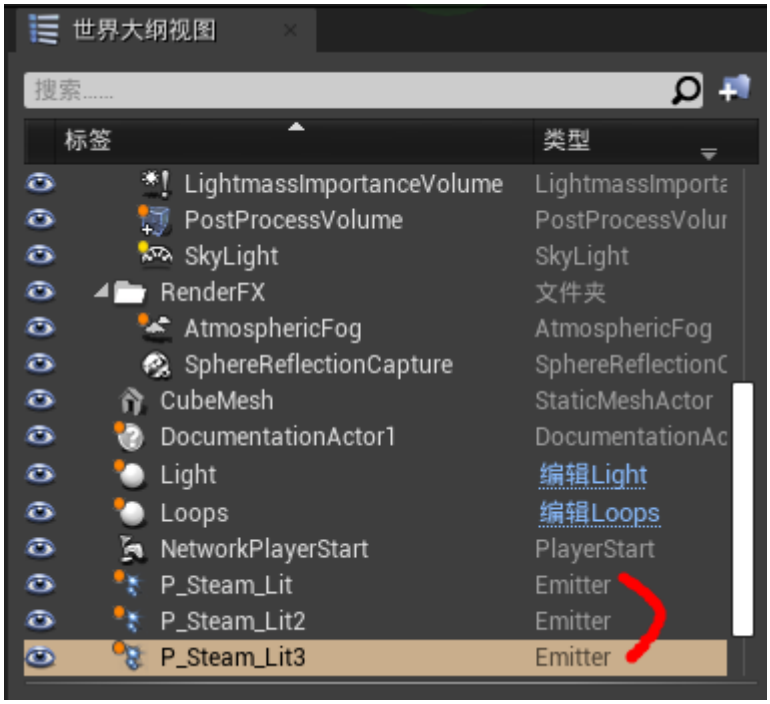
下面实现的功能是利用 ForEachLoop 循环切换同一类型 Actor 的显示状态。打开主界面中的内容浏览器，在内容面板中找到 StarterContent>>Particles>>P_Steam_Lit，这是烟雾粒子系统。



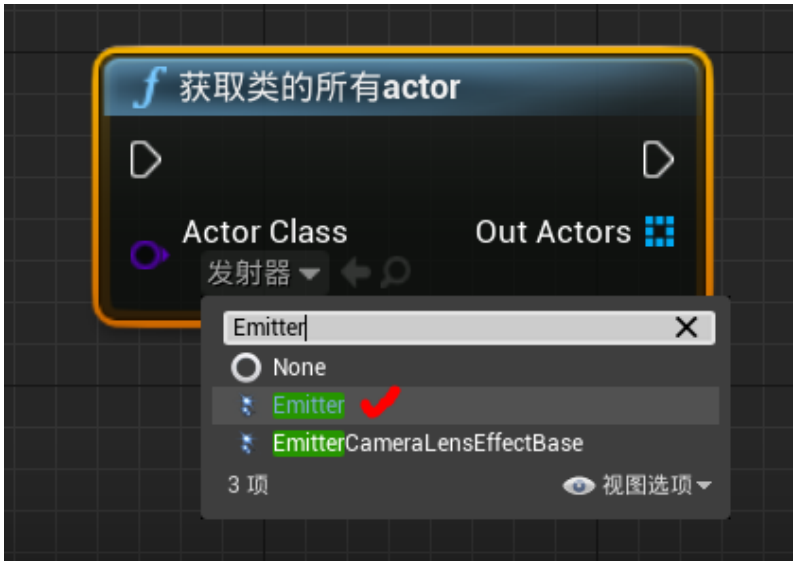
在场景中放置 3 个 P_Steam_Lit 粒子系统。



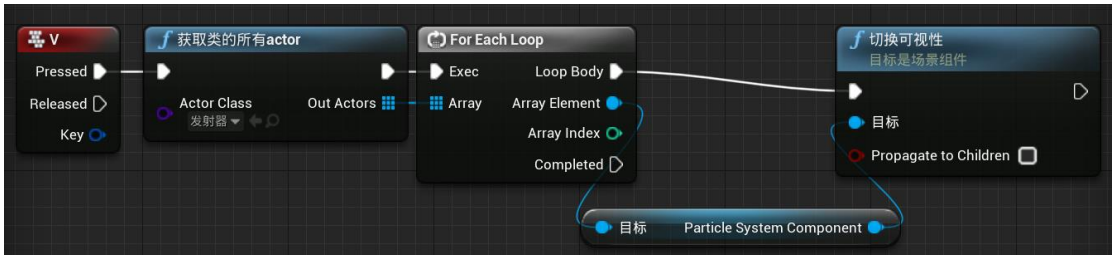
需要使用蓝图节点实现对这三个 P_Steam_Lit 粒子系统显示状态的切换。打开 Loops 蓝图，在事件图表编辑器中添加“获取类的所有 actor（Get All Actors of Class）”节点。把节点的 Actor Class 改为 P_Steam_Lit 所属的类。但如何能看到 P_Steam_Lit 资源对象所属的类？可以在主界面世界大纲视图中看到，这三个对象所属的类名为 Emitter。



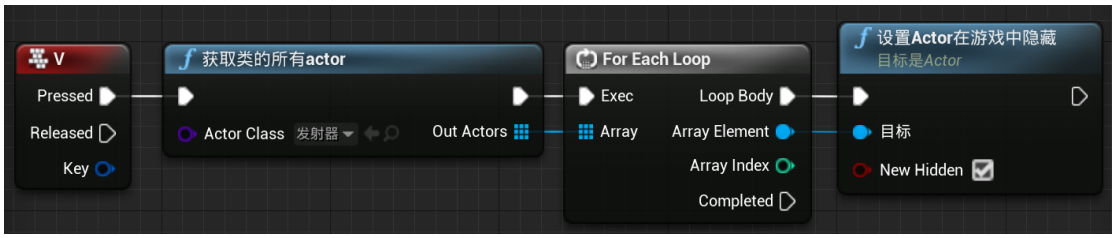
把“获取类的所有 actor”节点的 Actor Class 引脚改为 Emitter 类，选中后引脚会显示为“发射器”。



按照以往课程的内容，可以通过切换可视性节点来实现改变粒子系统的显示状态。把 Array Element 引脚拖出，找到“切换可视性”节点，会自动生成中间的类型转换节点。

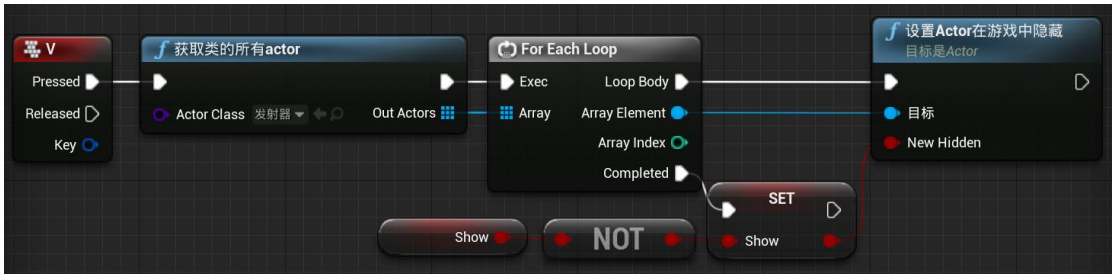


换一种方法来实现同样的功能。修改蓝图，添加“设置 Actor 在游戏中隐藏（Set Actor Hidden In Game）”节点，该节点功能是根据 New Hidden 引脚的值决定目标对象是隐藏（true）还是出现（false）在游戏中。



按目前的连接，游戏中按下 V 键，会隐藏所有 P_Steam_Lit 粒子系统对象。由于 New Hidden 是固定值 true，所以按多少次 V，都是隐藏对象，不会再次呈现。添加变量，布尔类型，命名为 Show，默认值为 true（勾选）。

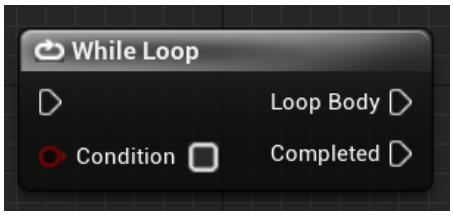
当循环结束时，修改 Show 的值（逻辑非），这样下次再按下 V 键时，就把更新后的 Show 值传给了“设置 Actor 在游戏中隐藏”节点中的 New Hidden 引脚。



在游戏中，按下 V 键后，会隐藏所有 P_Steam_Lit 粒子系统对象。再次按下 V 键后，所有被隐藏的 P_Steam_Lit 粒子系统对象会再次显现。

（4）WhileLoop 节点

WhileLoop 节点根据某个布尔值的真和假决定循环还是退出循环。在 Loops 蓝图事件图表编辑器中添加 WhileLoop 节点。



把判断是否循环的结果传给 WhileLoop 节点左侧 Condition 引脚。如果是 true 则循环，执行 Loop Body 引脚；如果是 false 则结束循环，执行 Completed 引脚。添加一个整数类型的数组变量，命名为 MyArray。不用修改默认值。

本节要实现的功能是向 MyArray 数据添加随机值，按 F 键后随机产生 20 个随机值并打印，打印完清空 MyArray 数组，以便下次按 F 键后重复上述操作。首先删除按键 F 节点后续的节点。

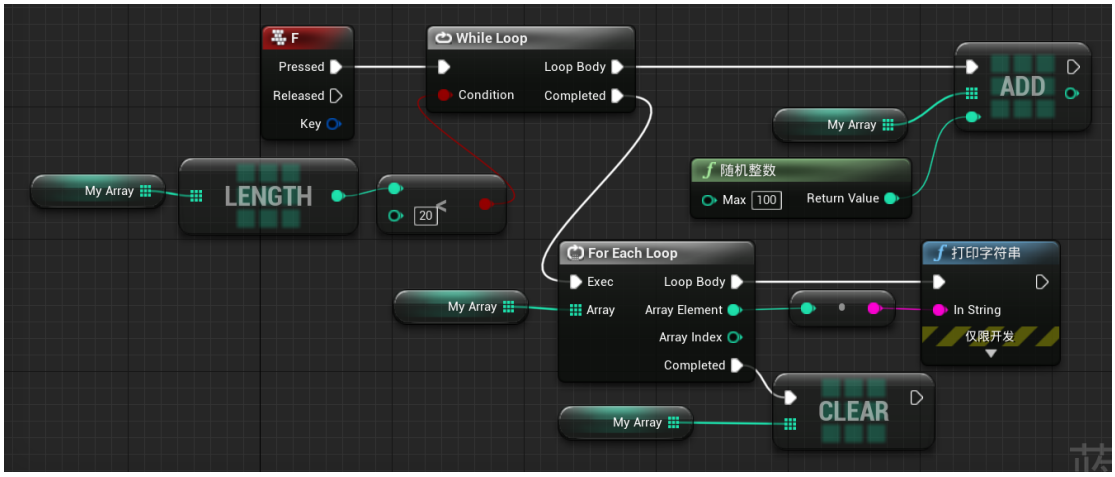
添加“工具集>>数组>>添加（Utilities>>Array>>Add）”节点（也可以通过“添加”搜索）。名字包含“添加”的节点非常多，注意要选择的是数组（Array）类型下的“添加”节点。

注意添加的“Add”节点左侧引脚应是数组类型。

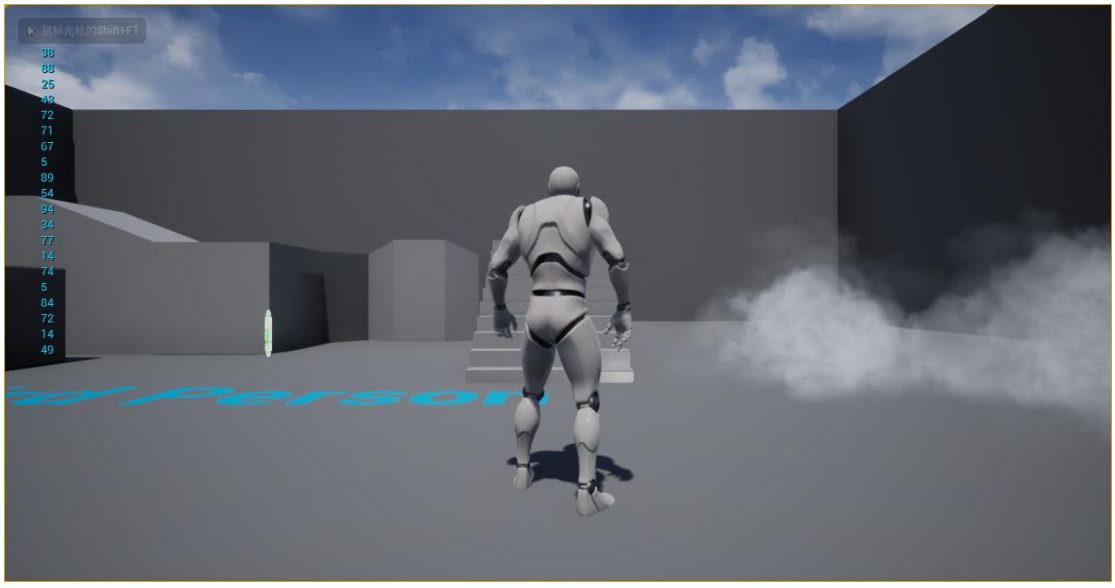
Add 节点可以向数组中添加一个元素（分别对应左侧两个数据引脚），将该元素插入到数组尾部，并调整数组的大小。Add 节点右侧有一个整数类型的数据引脚，表示当前添加的元素下标。

添加“随机整数（Random Integer）”节点，该节点用于产生 0 到 Max-1 范围内的随机整数。

把蓝图补充完整。当数组长度小于 20 时，向数据里添加一个 0 到 99 范围内的随机整数。添加完 20 个元素后，循环输出数组中的每一个元素值。输出完毕后清空数组。



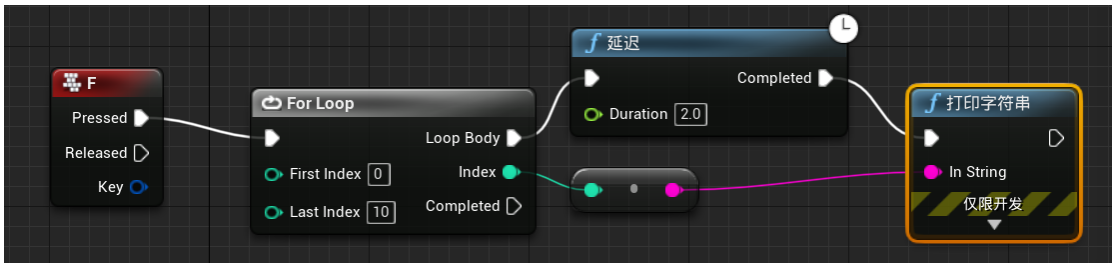
运行游戏时，当按下 F 键后，效果如图。



7、宏（Macro）

宏可以在类蓝图或者关卡蓝图中进行创建。本节将在类蓝图中自定义一个宏来实现伴有延迟的 WhileLoop 循环，并用它来实现带延迟的打印。

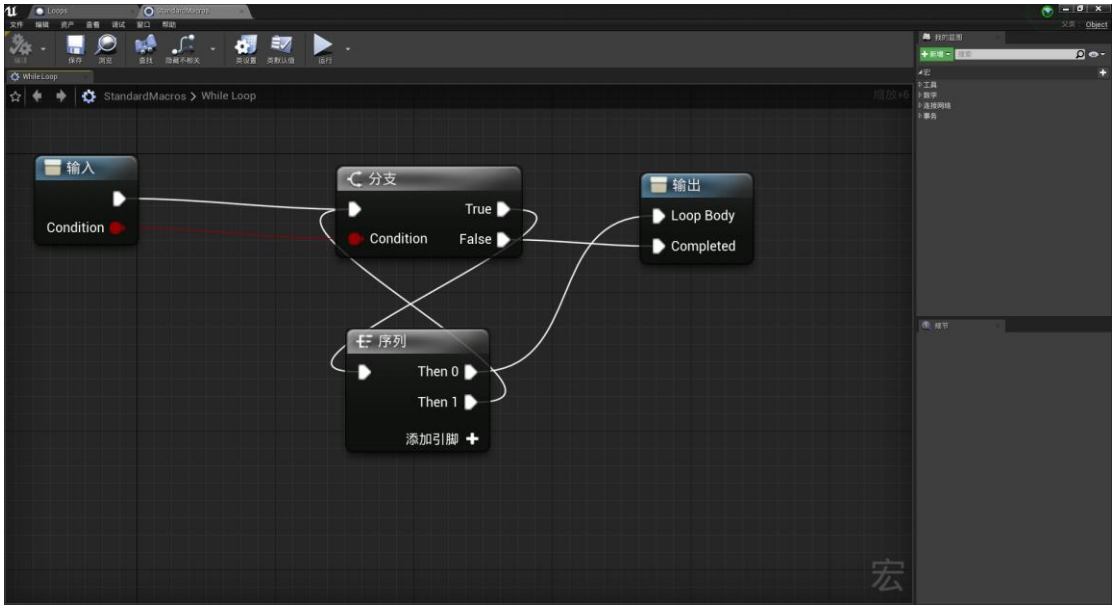
以下蓝图结构并不能实现带延迟的输出。



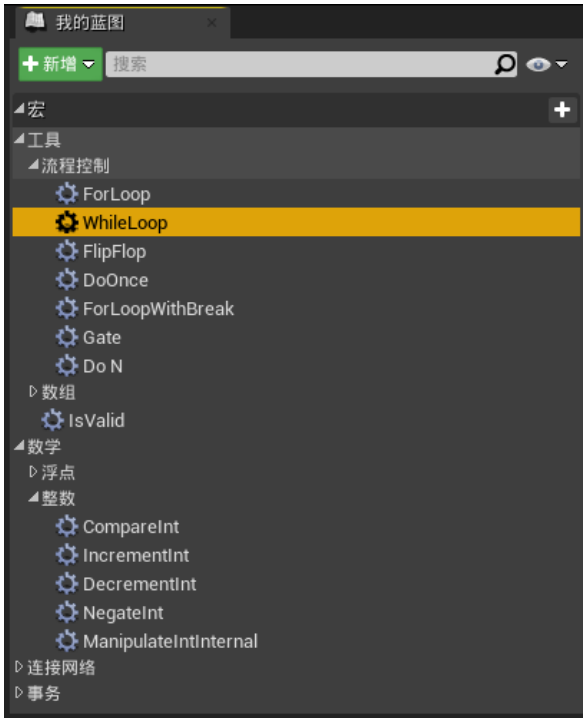
运行结果如下：

按下 F 键 2 秒钟后，屏幕上只输出了 11，而不是想象中的每隔 2 秒依次输出一个数字，分别是 0、1、…、10。为了实现延迟输出功能，需要了解宏。

首先观察系统已经实现的 WhileLoop 宏图表（只看别改）。打开 Loops 蓝图，找到在上一节中定义的 WhileLoop 节点，双击该节点，出现了 StandardMacros 蓝图宏库编辑器界面。

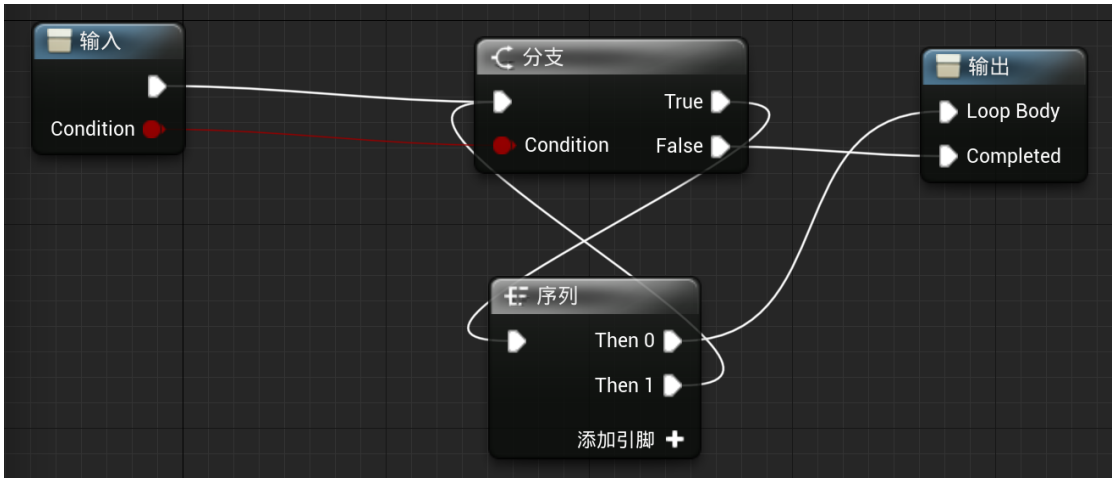


StandardMacros 为 WhileLoop 宏所在的蓝图宏库，在图表面板中可以看到 WhileLoop 宏节点布局。在图表面板右侧的“我的蓝图（My Blueprint）”面板中有一个“宏（Macro）”类目，它包含着 StandardMacros 蓝图宏库中的所有宏。



当选中其中某个宏时（例如 WhileLoop 宏），右下角的细节面板中就会呈现该宏的详细信息。

返回 WhileLoop 图表面板，可以看到 WhileLoop 宏由 4 个节点之间的关联来实现。其中“输入（Inputs）”输入通道节点和“输入（Outputs）”输出通道节点在宏中是默认存在的两个节点。“输入”节点中的 Condition 就是 WhileLoop 循环节点的输入数据引脚 Condition，“输出”节点中的 Loop Body 和 Completed 代表着 WhileLoop 循环节点右侧的输出执行引脚。



“序列（Sequence）”节点属于流程控制类型节点，当节点接收到输入执行信号时，将会按顺序执行右侧所有引脚。点击右侧“添加引脚（Add pin）”，可以继续添加想要的任意数量的输出节点。

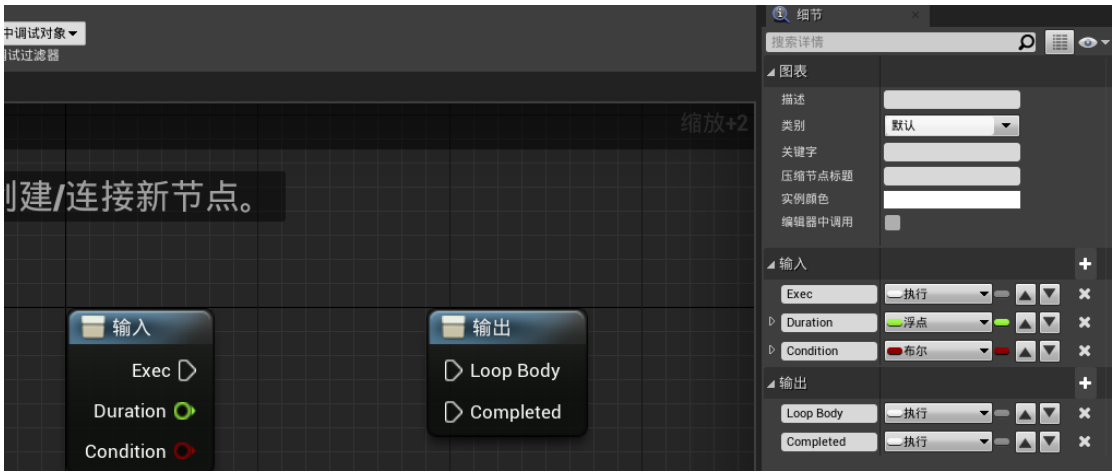
了解 WhileLoop 宏的运作原理，接下来要按照这个思路在 Loops 蓝图中创建一个用于执行延迟循环的宏。打开 Loops 蓝图，在“我的蓝图”面板中添加一个宏，命名为 WhileLoopWithDelay。

选中 WhileLoopWithDelay 宏，图表面板中默认有两个节点，分别是“输入”和“输出”节点。

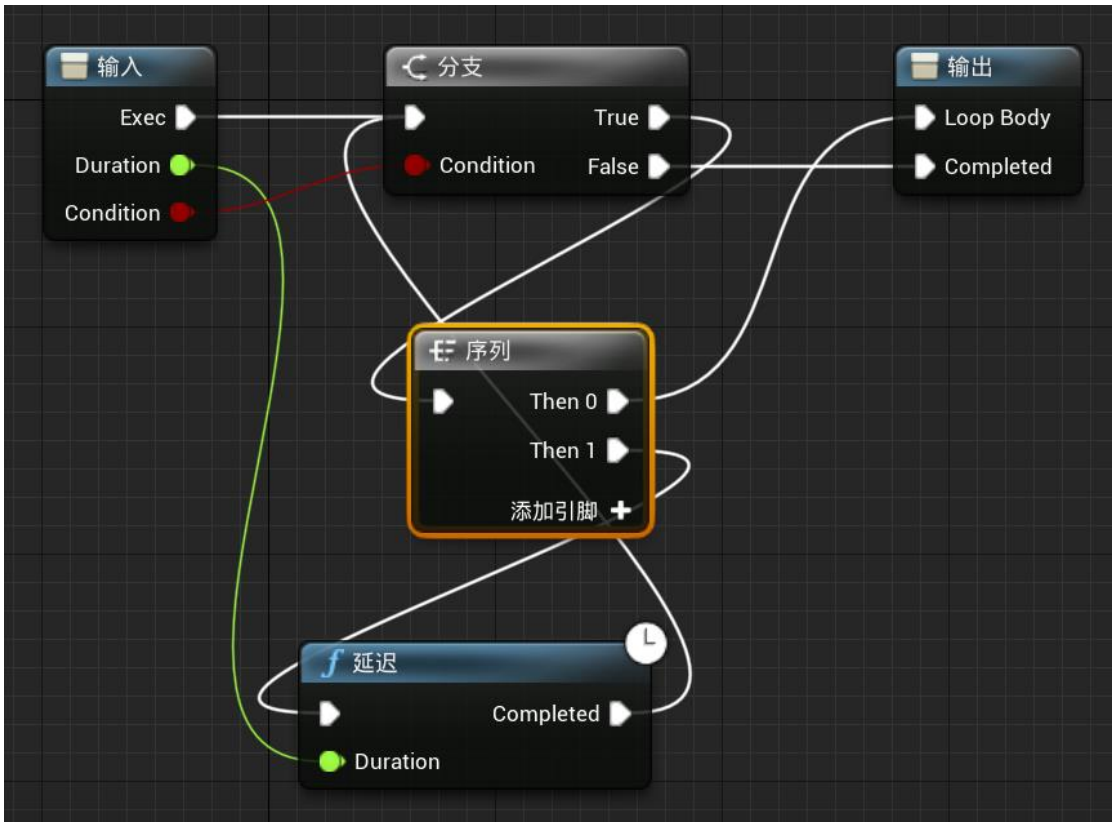
右侧的细节面板中会出现 WhileLoopWithDelay 宏的相关属性。

若要添加输入和输出执行引脚及数据引脚，需要单击细节面板中“输入”和“输出”类目下的“+”按钮，可以改变引脚名称和类型。

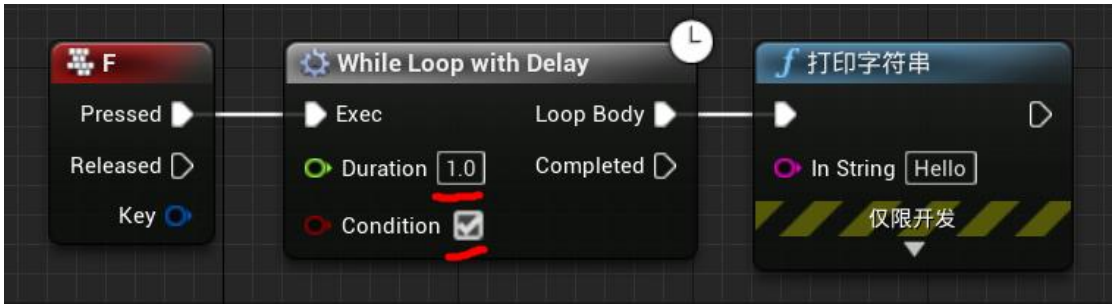
在“输入”中添加执行类型的引脚 Exec、浮点类型的引脚 Duration（延迟时间）、布尔类型的引脚 Condition，在“输出”中添加执行类型的引脚 Loop Body、执行类型的引脚 Completed。可以看到图表面板中输入和输出节点生成了相应的引脚。



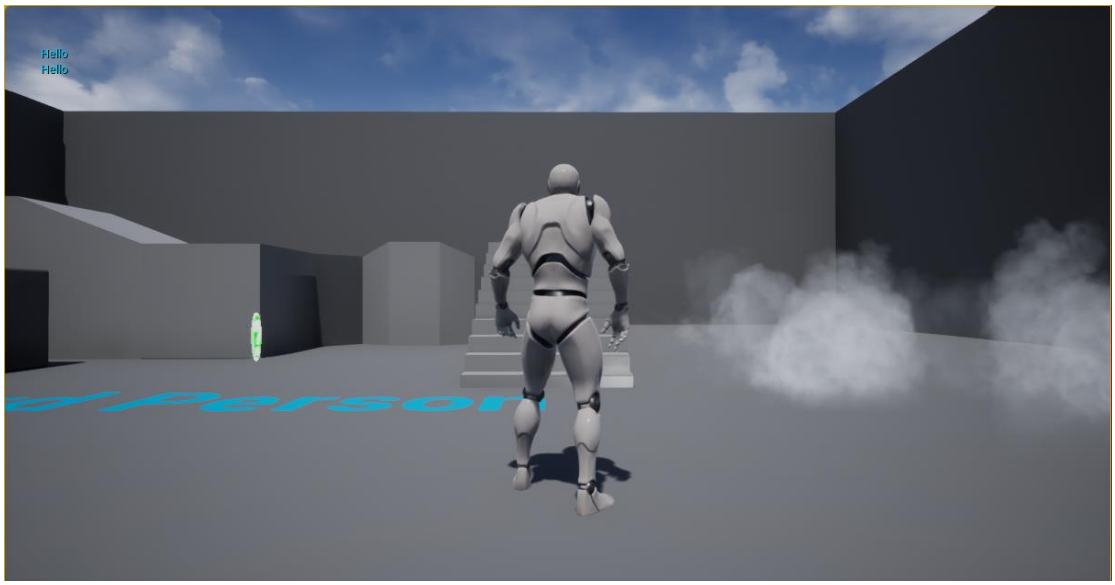
和 WhileLoop 宏一样，需要“分支（if）”节点和“序列（Sequence）”节点，还需要额外地加入“延迟（Delay）”节点。



保存并编译 WhileLoopWithDelay 宏，现在利用 WhileLoopWithDelay 节点打印字符串。返回 Loops 蓝图的事件图表编辑器，添加 WhileLoopWithDelay 节点，设置 Duration 参数为 1，表示循环间的时间间隔为 1s；勾选 Condition 参数将它设为 true，即循环条件始终成立。修改按键 F 节点的连接。



运行游戏后，按 F 键，效果如图，每隔 1 秒屏幕上输出一个 Hello。



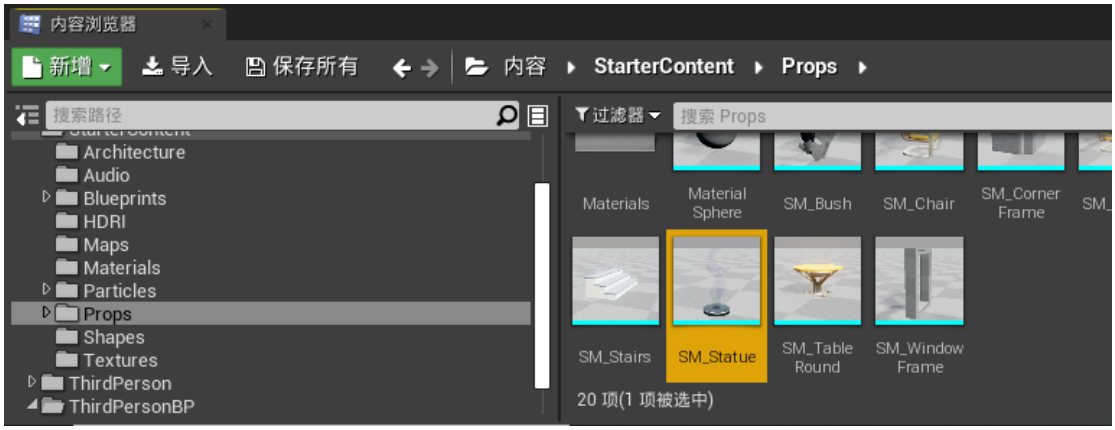
8、Spawn 类型节点

Spawn 节点可以生成指定的实例对象。本节将介绍“生成发射器（Spawn Emitter）”节点和“生成角色（Spawn Actor）”节点以及变量的“生成时公开（Expose On Spawn）”属性。

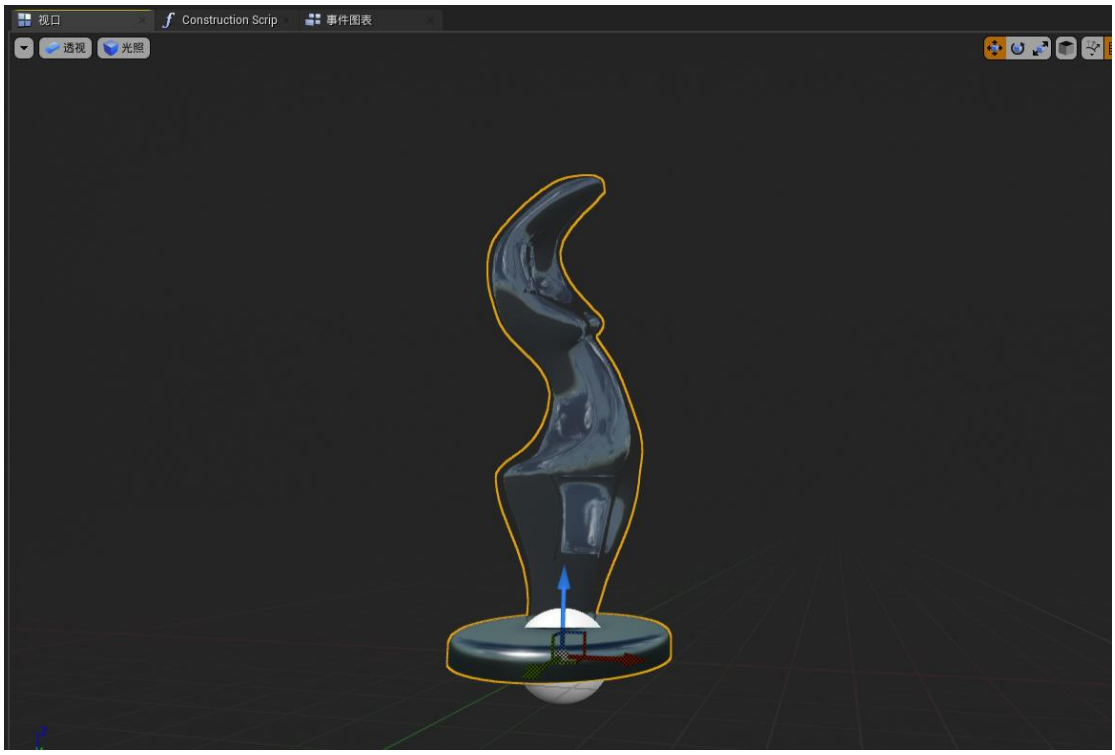
（1）“生成发射器（Spawn Emitter）”节点

Spawn Emitter 节点可以在指定位置产生 Emitter 类型实例对象（例如上一节中的烟雾粒子系统）。本节将实现接触指定物体后产生爆炸的效果。

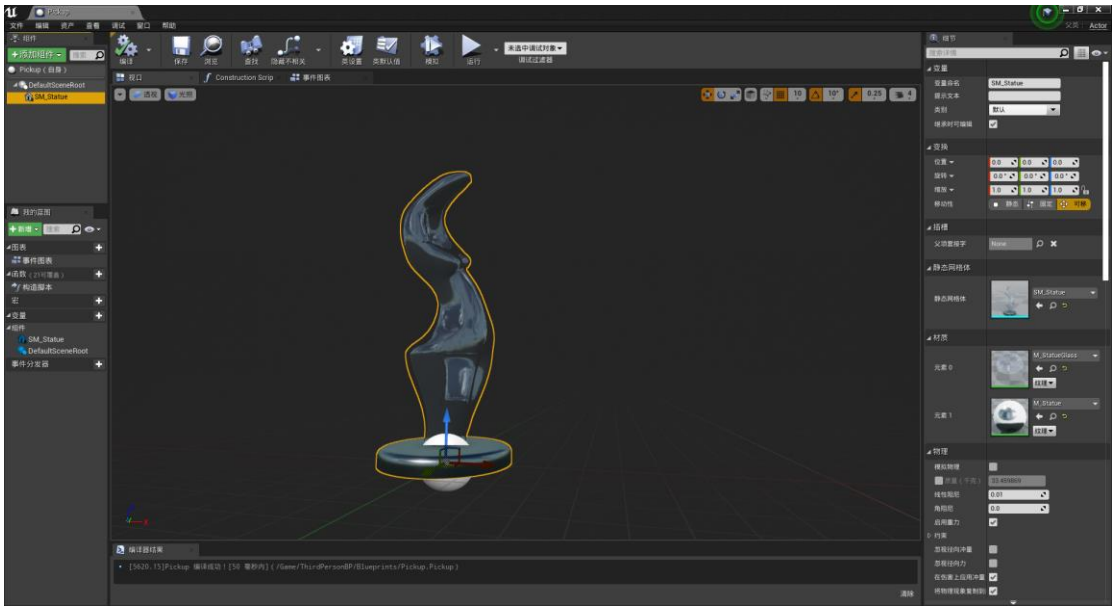
沿用之前的工程项目，创建一个以 Actor 类为父类的蓝图，命名为 Pickup。打开 Pickup 蓝图，切换到视口面板。在主界面内容浏览器中，找到 SM_Statue 静态模型（StarterContent>>Props 文件夹）。

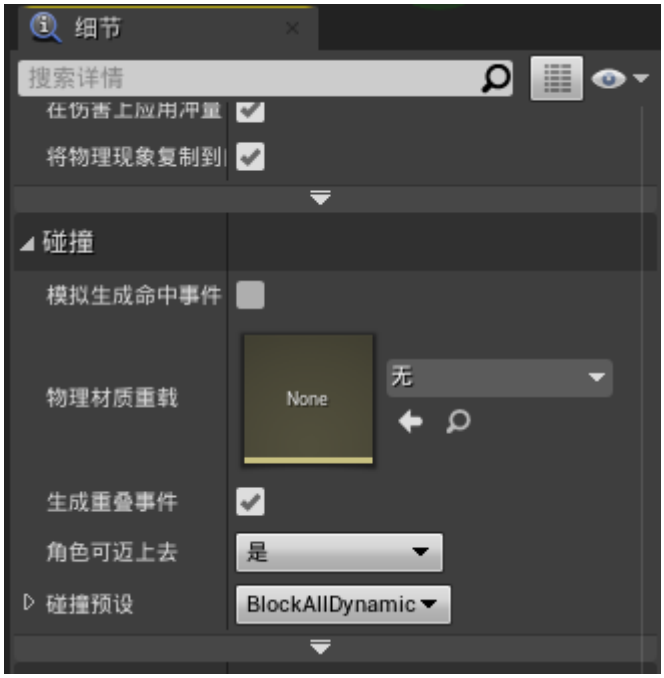


拖动 SM_Statue 静态模型到 Pickup 蓝图的视口面板中，从而在 Pickup 蓝图中添加了一个 SM_Statue 静态模型组件。



选中 SM_Statue 组件，可以在右侧细节面板中找到“碰撞”类目。





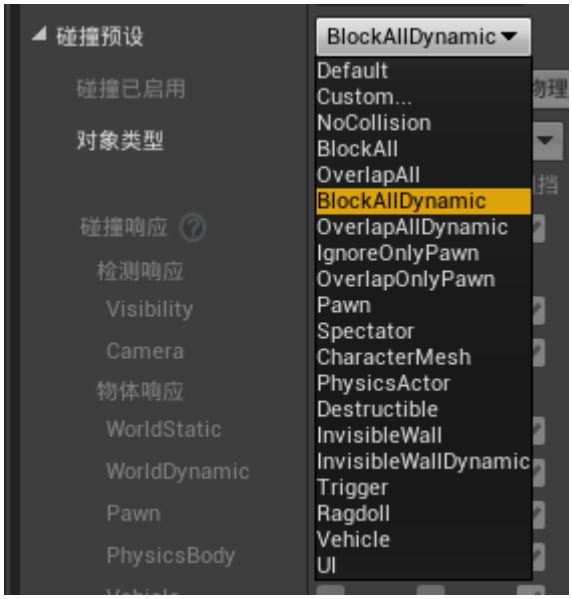
展开其中的“碰撞预设”类目，可以看到有许多碰撞属性。

比较常用的碰撞属性有生成重叠事件（Generate Overlap Events）、碰撞预设（Collision Presets）和碰撞响应（Collision Responses）。

（a）生成重叠事件：如果想让一个对象生成重叠事件，如 ActorBeginOverlap 事件、ActorEndOverlap 事件，相关的节点如前几节的 OnActorBeginOverlap 节点、OnActorEndOverlap 节点，则该标志需要设置为真。

（b）碰撞预设：包含碰撞启用、对象类型和碰撞响应项的一组预设值。

关于碰撞预设值，有很多选择。



每种碰撞预设对应各种常见类型的对象。

属性	说明
默认（Default）	使用已在静态网格体编辑器中应用给静态网格体的设置。
自定义...（Custom...）	为此实例设置所有自定义碰撞设置。

NoCollision	无碰撞。
BlockAll	在默认情况下阻挡所有 Actor 的 WorldStatic 对象。所有新自定义信道都将使用其本身的默认响应。
OverlapAll	在默认情况下与所有 Actor 重叠的 WorldStatic 对象。所有新自定义信道都将使用其本身的默认响应。
BlockAllDynamic	在默认情况下阻挡所有 Actor 的 WorldDynamic 对象。所有新自定义信道都将使用其本身的默认响应。
OverlapAllDynamic	在默认情况下与所有 Actor 重叠的 WorldDynamic 对象。所有新自定义信道都将使用其本身的默认响应。
IngoreOnlyPawn	忽略 Pawn 和载具的 WorldDynamic 对象。所有其他信道都将设置为默认值。
OverlapOnlyPawn	与 Pawn、摄像机和载具重叠的 WorldDynamic 对象。所有其他信道都将设置为默认值。
Pawn	Pawn 对象。可用于任意可操作角色或 AI 的胶囊体。
Spectator	忽略除 WorldStatic 以外的所有其他 Actor 的 Pawn 对象。
CharacterMesh	用于角色网格体的 Pawn 对象。所有其他信道都将设置为默认值。
PhysicsActor	模拟 Actor。
Destructible	可破坏物 Actor。
InvisibleWall	不可见的 WorldStatic 对象。
InvisibleWallDynamic	不可见的 WorldDynamic 对象。

Trigger	用于触发器的 WorldDynamic 对象。所有其他信道都将设置为默认值。
Ragdoll	模拟骨架网格体组件。所有其他信道都将设置为默认值。
Vehicle	阻挡载具（Vehicle）、WorldStatic 和 WorldDynamic 的载具对象。所有其他信道都将设置为默认值。
UI	在默认情况下与所有 Actor 重叠的 WorldStatic 对象。所有新自定义信道都将使用其本身的默认响应。

（c）碰撞响应：是否同物理资源中的特定物理刚体发生碰撞，包括检测响应（Trace Responses）和物体响应（Object Responses）两类。“检测响应”用于追踪光线投射，例如“按信道进行线迹追踪（Line Trace by Channel）”蓝图节点。“物体响应”定义了一个对象可以简单忽略、重叠或者阻挡同它交互的各类对象。碰撞响应包含以下功能选项的设置。

忽略（Ignore）：无论另一个物体的“碰撞响应（Collision Responses）”为何值，此物体都将忽略交互。

重叠(Overlap)：如果已将另一个物体设置为“重叠(Overlap)”或“阻挡(Block)”此物体将发生重叠事件（Overlap)。

阻挡（Block)：如果已将另一个物理形体设置为“阻挡（Block）”此物理形体的“对象类型（Object Type）”，将发生撞击事件（Hit Event)，不触发重叠事件（Overlap)。

Visibility：当被可视性测试信道检测到时，此物体做出如何反应。例如当碰撞预设为“CharacterMesh”或者“Pawn”时，Visibility 项为“忽略”，原因是当角色向外发出检测光线时（例如遥控开空调)，得无视自己，否则检测光线会先碰撞到自身，从而被自身阻挡，无法检测到目标。

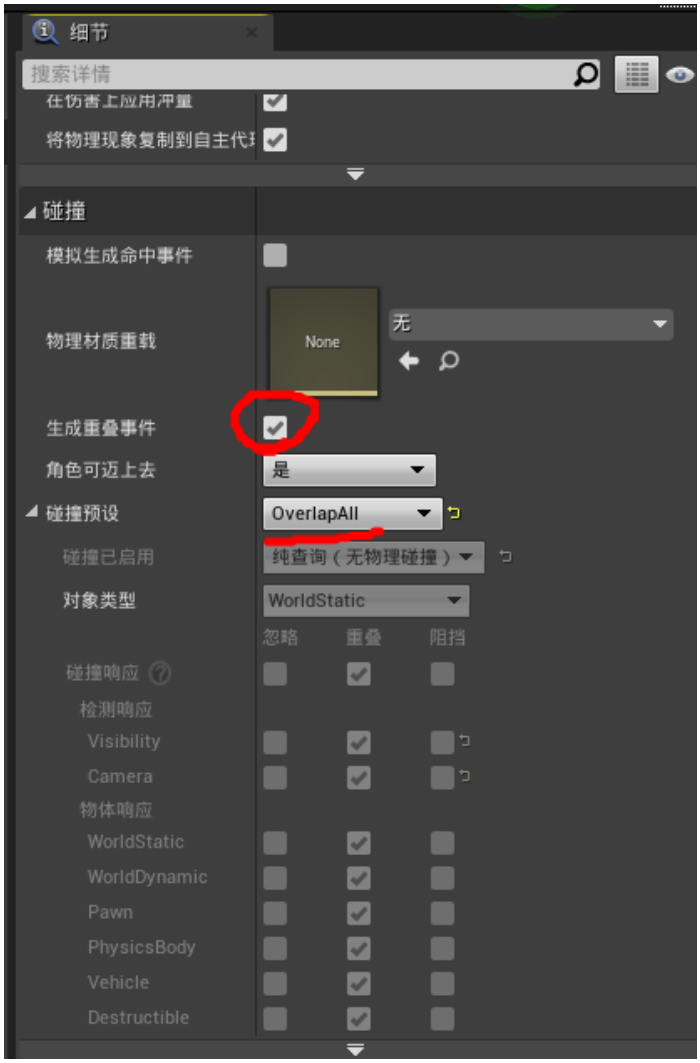
Camera：当被摄像机发出光线检测到时，此物体做出如何反应。

WorldStatic：当与 WorldStatic 类型物体对象交互时，此物体做出如何反应。（其他物体响应选项类似，只是换作与另一种类型的物体对象交互）

关于这几类物体的描述如下：

类型	说明
WorldStatic	静态的 Actor
WorldDynamic	可移动的 Actor
Pawn	角色
PhysicsBody	物理刚体
Vehicle	车辆载具
Destructible	可破坏的 Actor

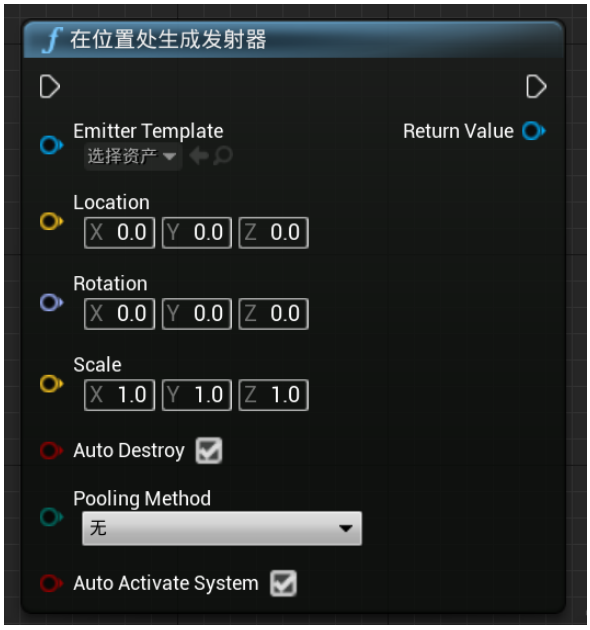
了解以上关于碰撞的知识后，回到 Pickup 蓝图，把 SM_Statue 组件的碰撞预设值修改为 OverlapAll，确认上方的“生成重叠事件”被勾选。



如果没有选中“生成重叠事件”，那么在碰撞响应中选择“重叠”和选择“忽略”是没有区别的。

完成以上准备工作，就可以开始设计蓝图。切换到事件图表编辑器面板，先选中 SM_Statue 组件，然后添加“组件开始重叠时（Add On Component Begin Overlap）”节点。

当 SM_Statue 组件与其他 Actor 对象之间产生重叠时，会执行“组件开始重叠时(SM_Statue)”事件节点。重叠发生后要产生爆炸效果，因此接下来要执行的是在 SM_Statue 所在位置生成有爆炸效果的粒子系统，需要使用“在位置处生成发射器（Spawn Emitter at Location）”节点。



“在位置处生成发射器”节点左侧的参数中，Emitter Template 可以选择粒子系统类的资源对象（例如爆炸粒子系统），Location 指定产生资源的世界坐标位置，Rotation 指定资源的旋转角度，Scale 指定资源的缩放值，Auto Destroy 表示粒子系统效果播放完毕后是否自动销毁。

单击 Emitter Template 选择资产 P_Explosion。

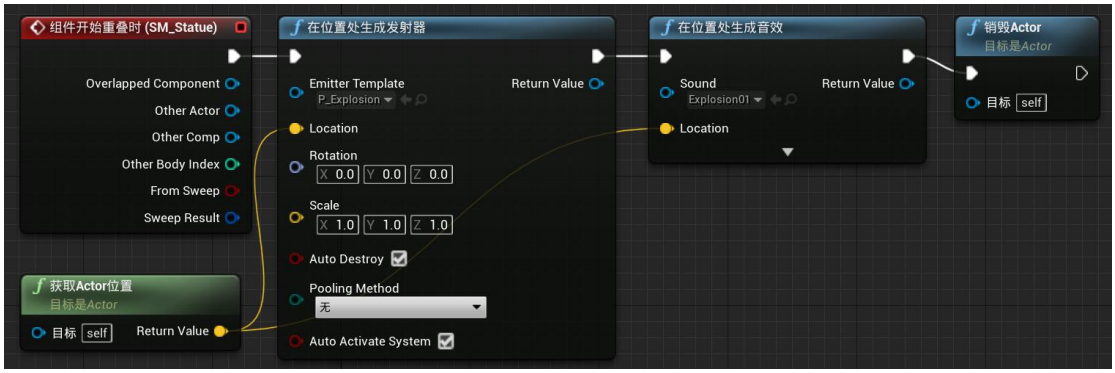
添加“在位置处生成音效（Spawn Sound at Location）”节点，该节点可以在指定位置处播放音效。选择 Sound 为 Explosion01。

添加“获取 Actor 位置（Get Actor Location）”节点，其中目标值采用默认的

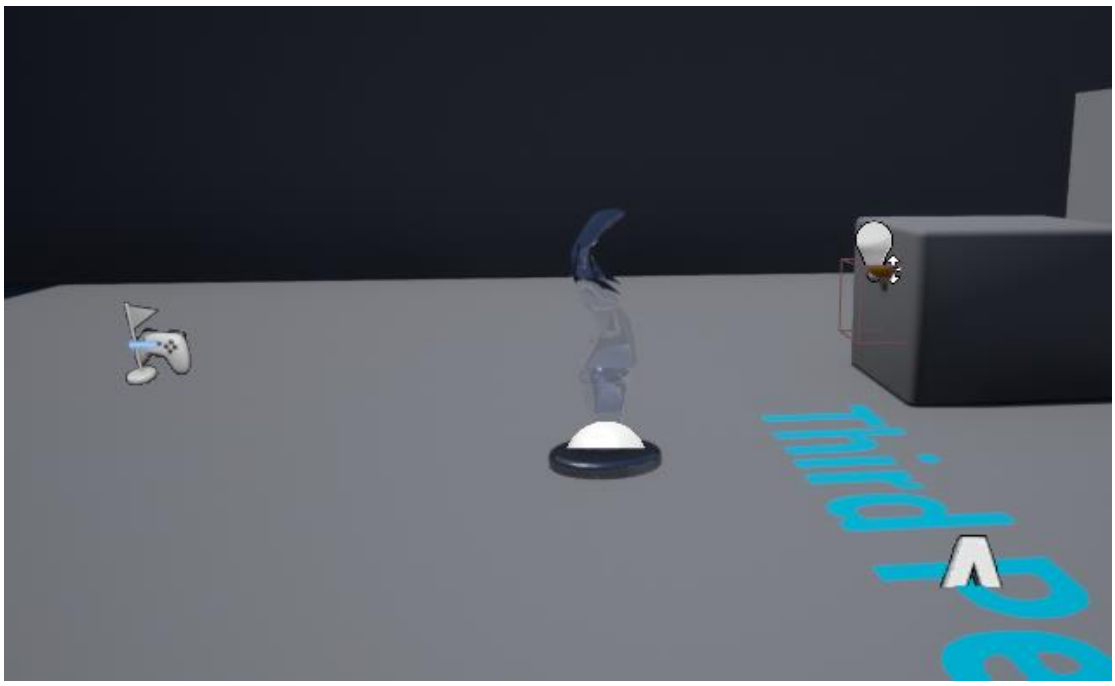
Self，表示是获取当前 Actor 的位置，即 SM_Statue 组件所属的 Pickup 蓝图类对象。

添加“销毁 Actor（Destroy Actor）”节点，其中目标值采用默认的 Self，表示该节点用于销毁当前 Actor 类的对象。

连接所有节点。

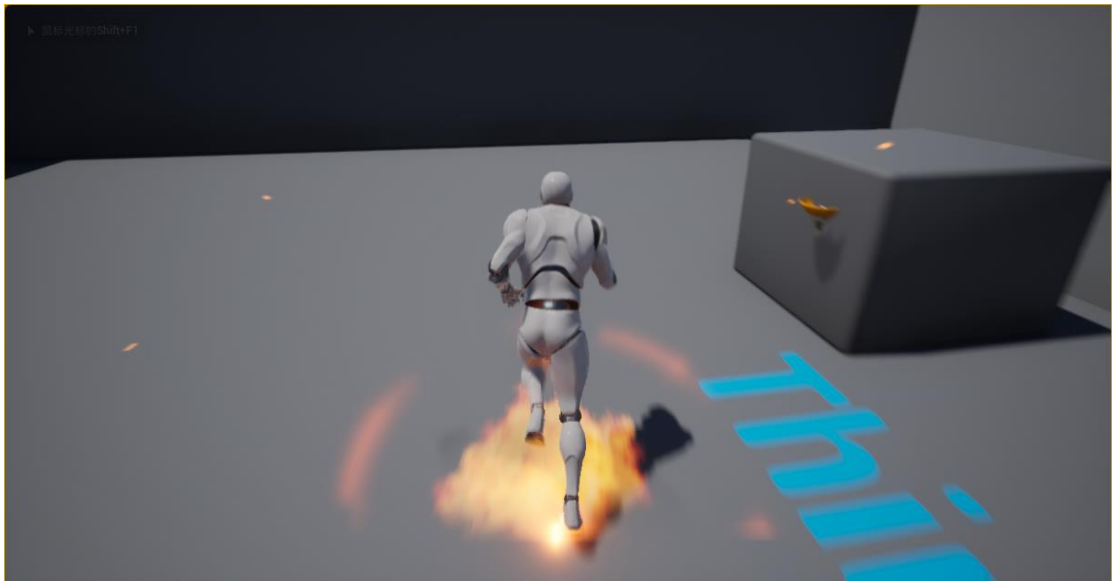


把 Pickup 类拖入主界面场景中。



运行游戏，当玩家移动至雕像附近时，产生爆炸粒子效果和音效，雕像被销

毁。



作业：根据“第 7 课作业效果.mp4”的效果设计蓝图。