第七周

蓝图和关卡蓝图

自我介绍

- 蚂蚁集团.体验技术部.高级技术专家
- 多年游戏从业经验
  - 主导研发三代自研引擎，发布产品：
    - 《仙剑3》、《功夫世界》、《龙Online》
  - 现在主攻虚幻引擎技术
    - 国内首批**虚幻3** MMORPG：《神兵传奇》
    - 基于**虚幻4**的数据可视化中台系统

房燕良

本周内容

课程内容

- 为什么要学习蓝图?
- 什么是蓝图?
- 蓝图与C++
- 创建蓝图、蓝图编辑器
- 蓝图的节点、引脚与引线
- 蓝图的变量、数据类型与数据结构
- 蓝图的运算符和函数

学习成果

- 对于蓝图作为一种可视化脚本语言有一个全面的理解
- 能够制作简单的具有可玩性的蓝图

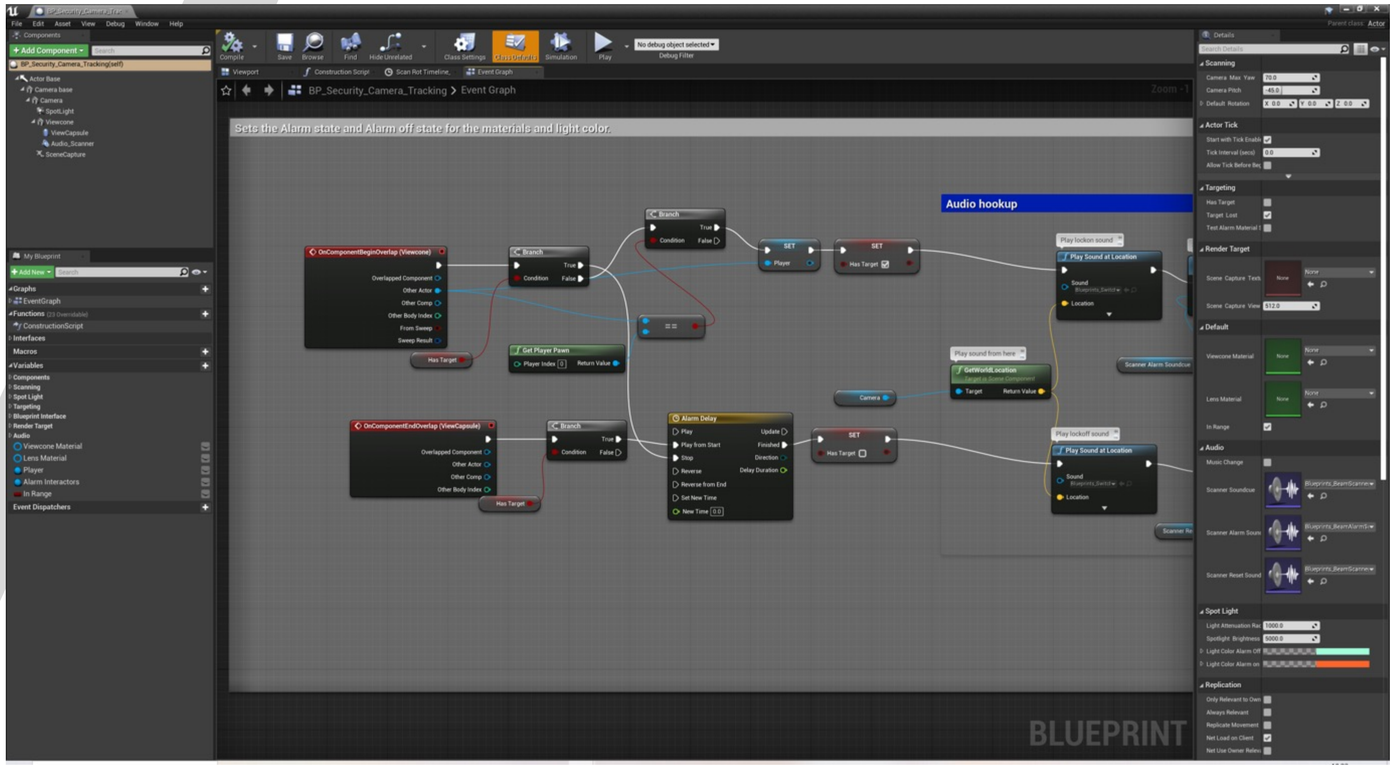
为什么要学习蓝图？

Motivation

为什么要学习蓝图？

蓝图作为虚幻4的基础设施，涉及到多个系统：

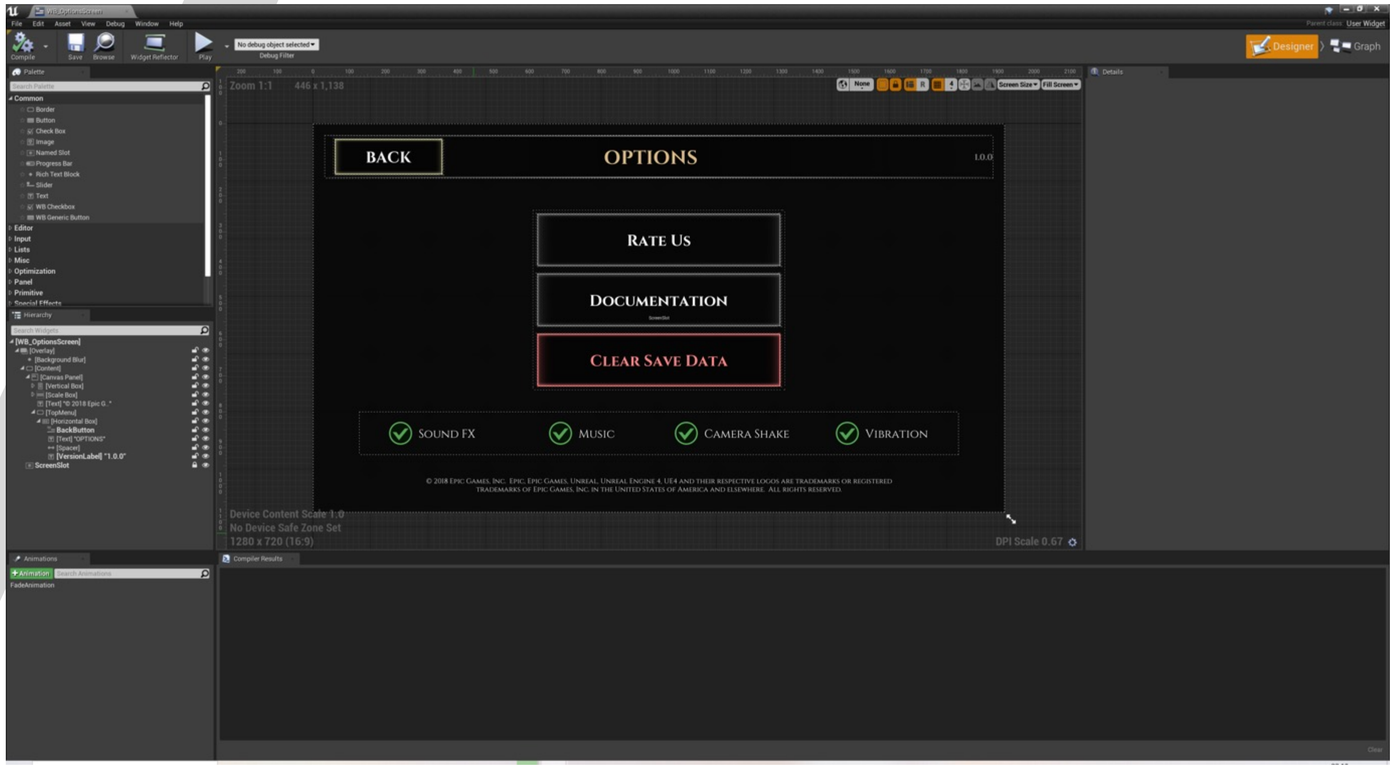
- **Gameplay**
- UMG Widget Blueprint
- Animation Blueprint
- Unreal Editor Utility Blueprints
- … …



为什么要学习蓝图？

蓝图作为虚幻4的基础设施，涉及到多个系统：

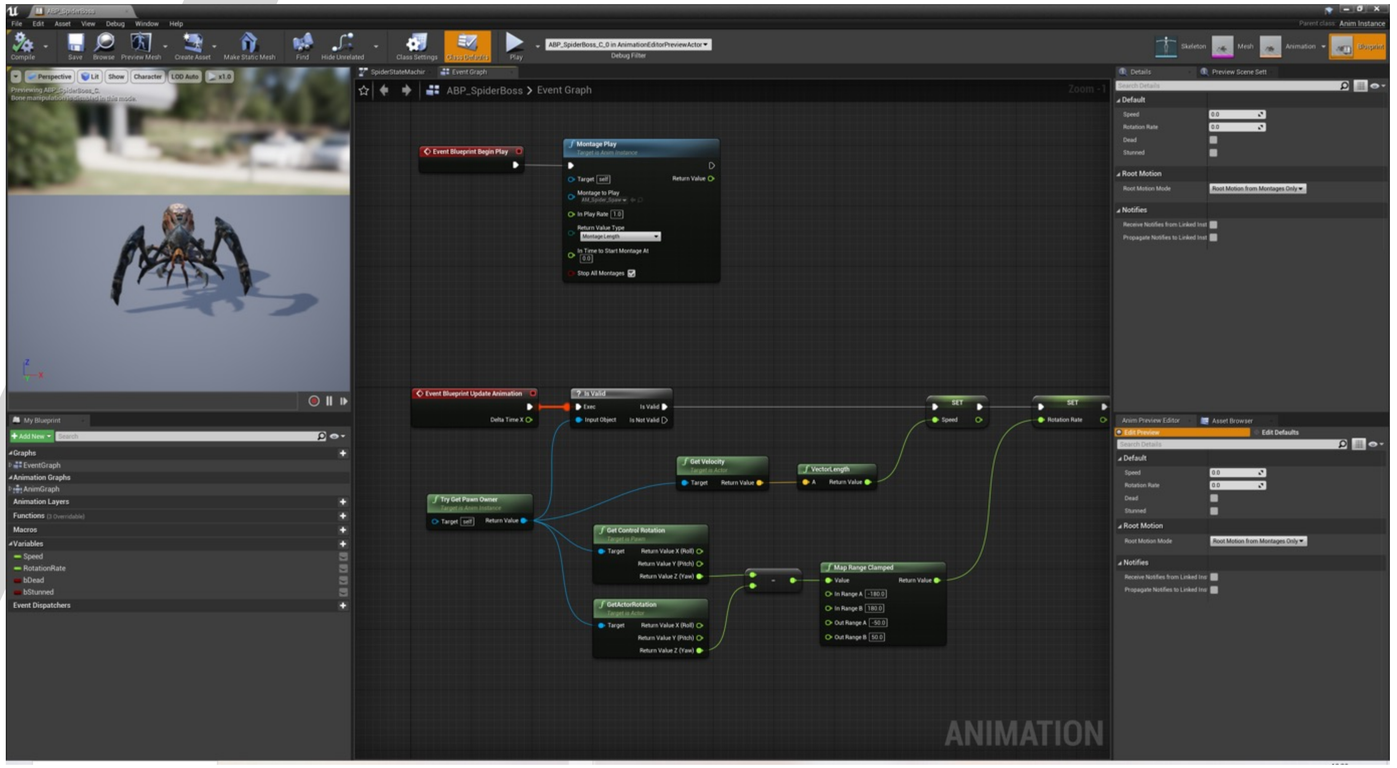
- Gameplay
- **UMG Widget Blueprint**
- Animation Blueprint
- Unreal Editor Utility Blueprints
- ……



为什么要学习蓝图？

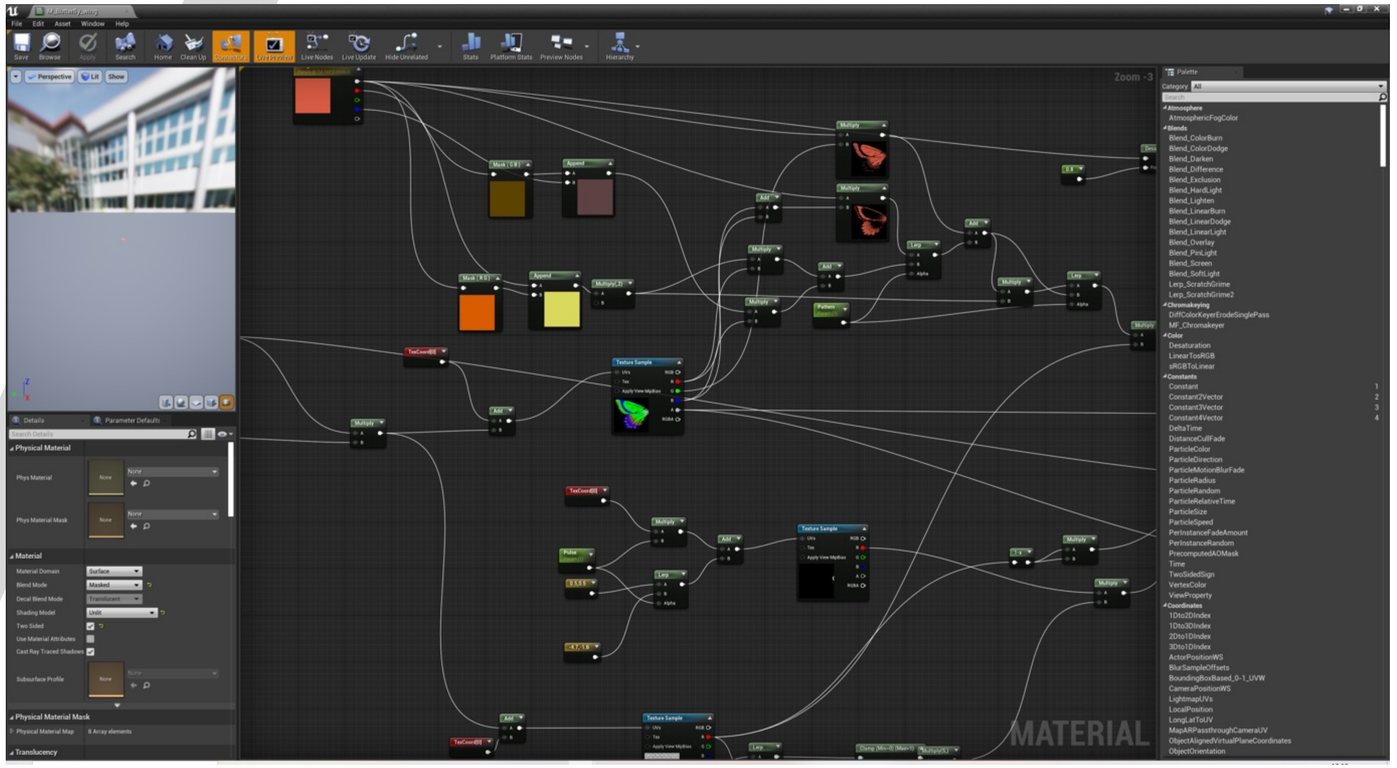
蓝图作为虚幻4的基础设施，涉及到多个系统：

- Gameplay
- **UMG Widget Blueprint**
- Animation Blueprint
- Unreal Editor Utility Blueprints
- ……



顺带提一下

- 虚幻引擎中还有一些长得很像蓝图的 Graph
- Material Graph 不是蓝图

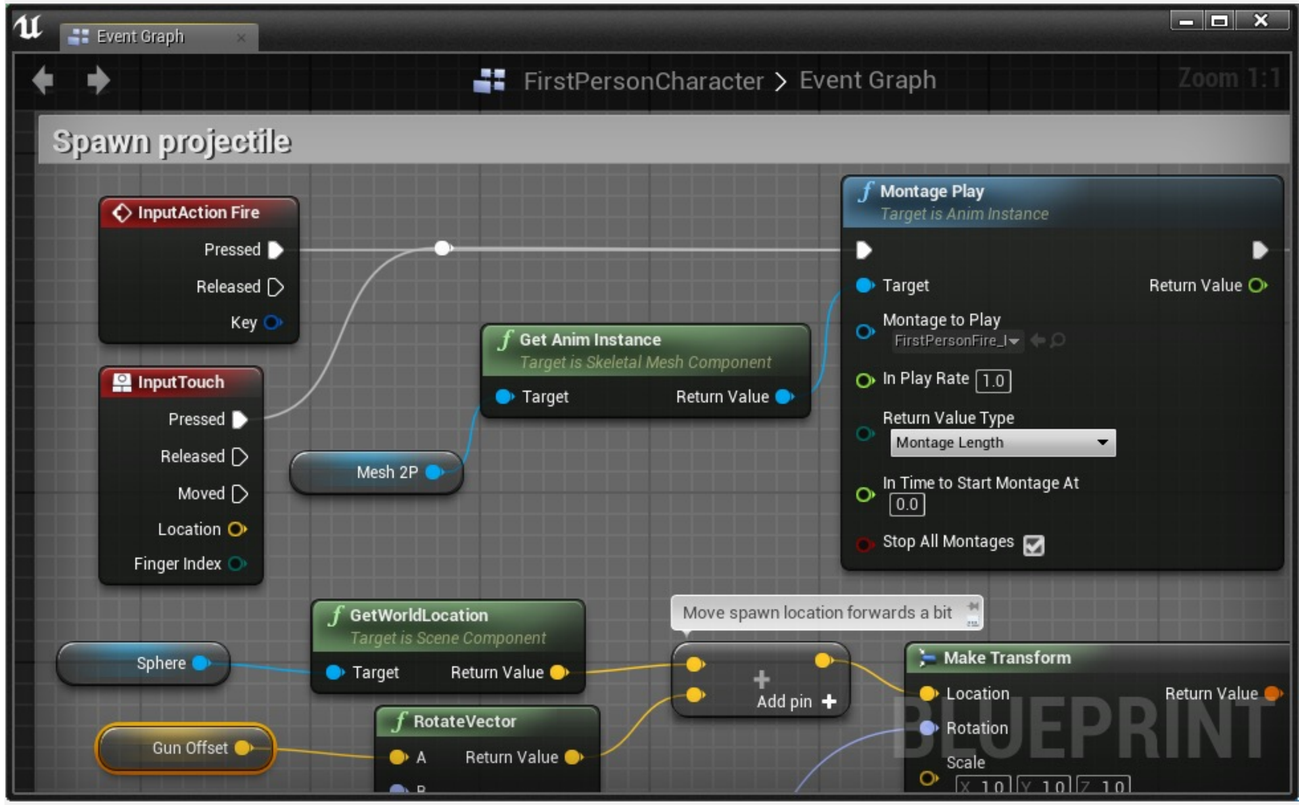


什么是蓝图？

What is Blueprint?

什么是蓝图?

- 蓝图(Blueprint)是 Epic Games 针对虚幻4引擎开发的可视化脚本语言(Visual Scripting)
- “蓝图”一词也用于指代使用蓝图创建的游戏对象



主要蓝图类型

关卡蓝图(Level Blueprint)

关卡蓝图是一种特殊类型的蓝图，属于关卡。它用于定义关卡中的特定事件和操作。

关卡蓝图可以用于与蓝图Actor类互动，以及管理某些系统，如过场动画和关卡流送。

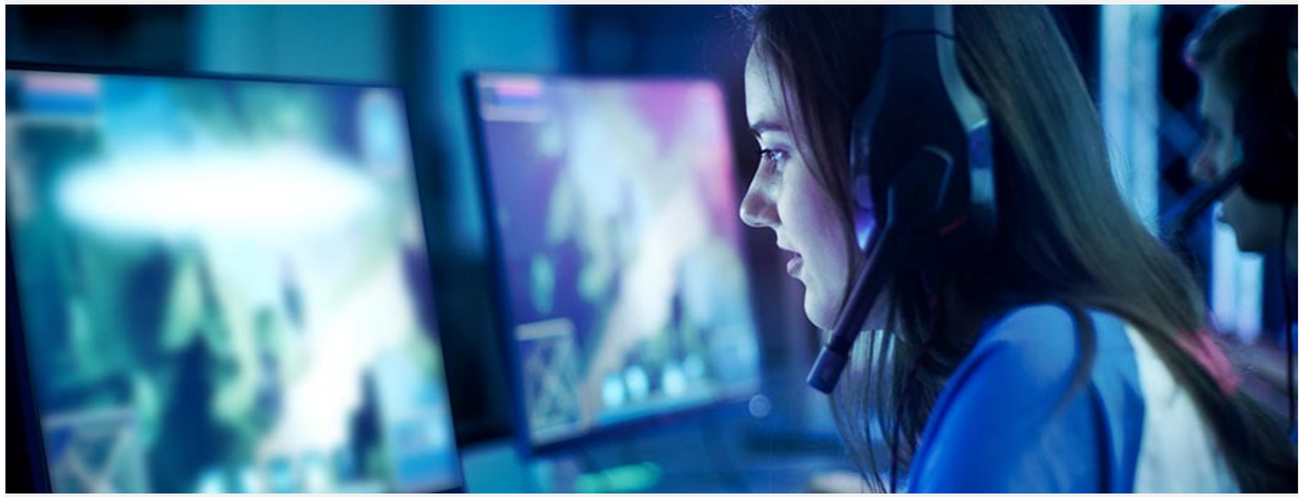
蓝图类(Blueprint Class)

类是对特定类型对象使用的数据和行为的定义。蓝图类可以基于C++类或另一个蓝图类。

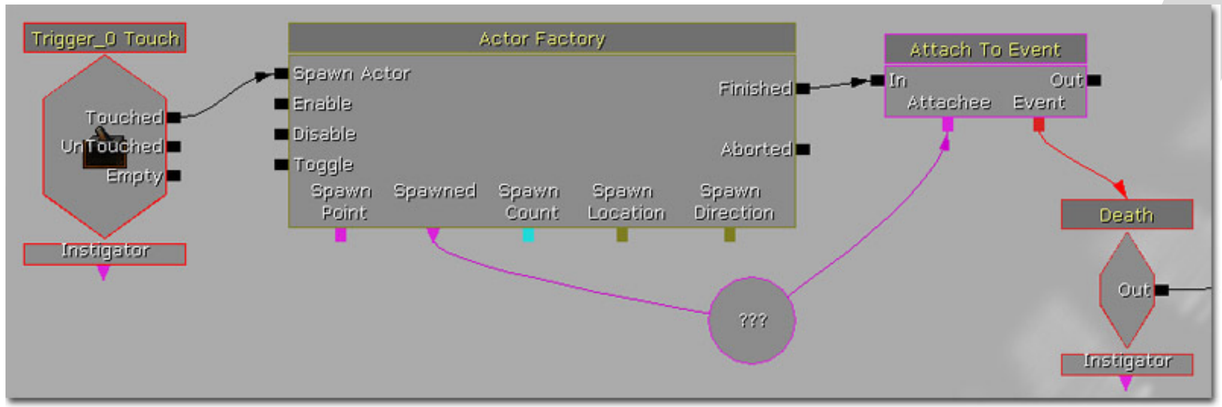
蓝图类用于为游戏创建互动对象，并可以在任意关卡中重复使用。

蓝图的设计理念

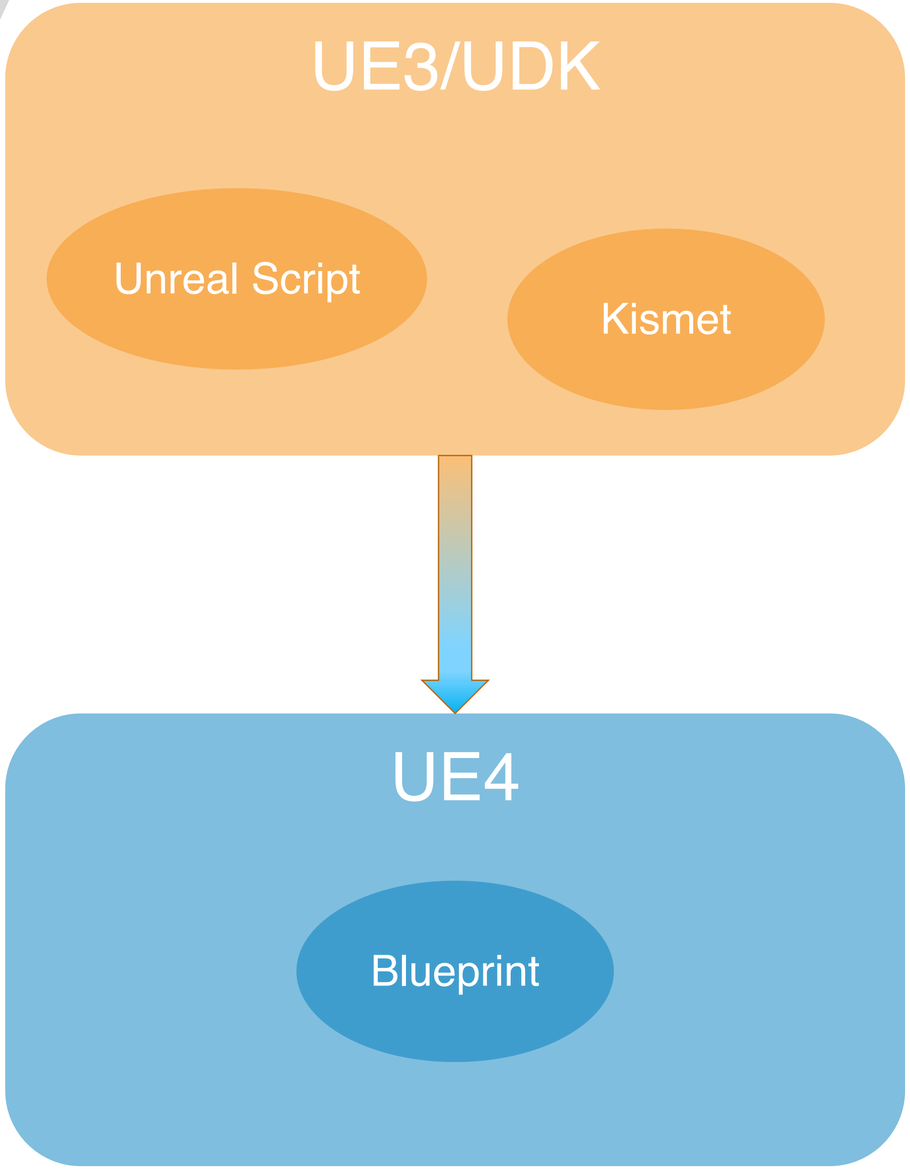
- 对于整个团队来说，都是易用的、友好的
- 不要求使用者有编程背景
- 能够帮助团队进行快速迭代



蓝图的前世今生



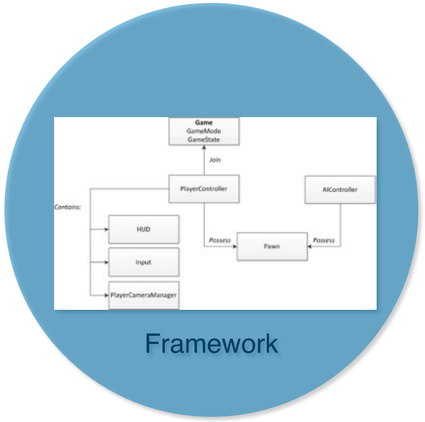
* https://docs.unrealengine.com/udk/Three/KismetUserGuide.html



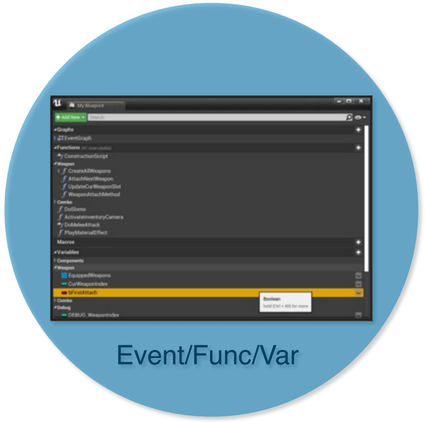
蓝图与C++

Blueprint and C++

引擎使用C++开发



Framework

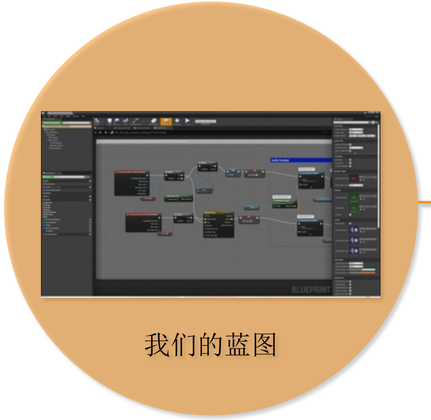


Event/Func/Var

Function Library

继承&派生

依赖&调用



我们的蓝图

编译&运行

蓝图与C++

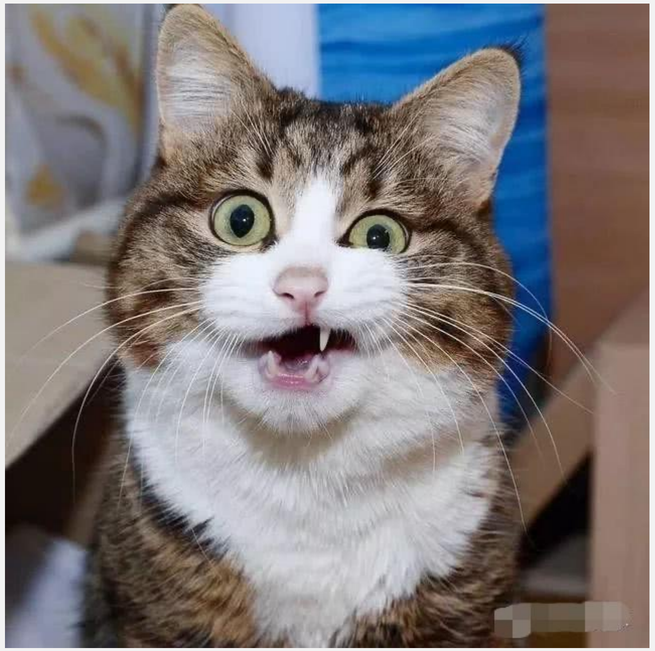
- 虚幻4中的两种 C++ 编程：Native C++、Unreal C++
- Unreal C++ 和 蓝图 处在同一层级
- 性能等深入的话题留到下一周

Hello，world！

第一个蓝图类

蓝图是一种脚本语言

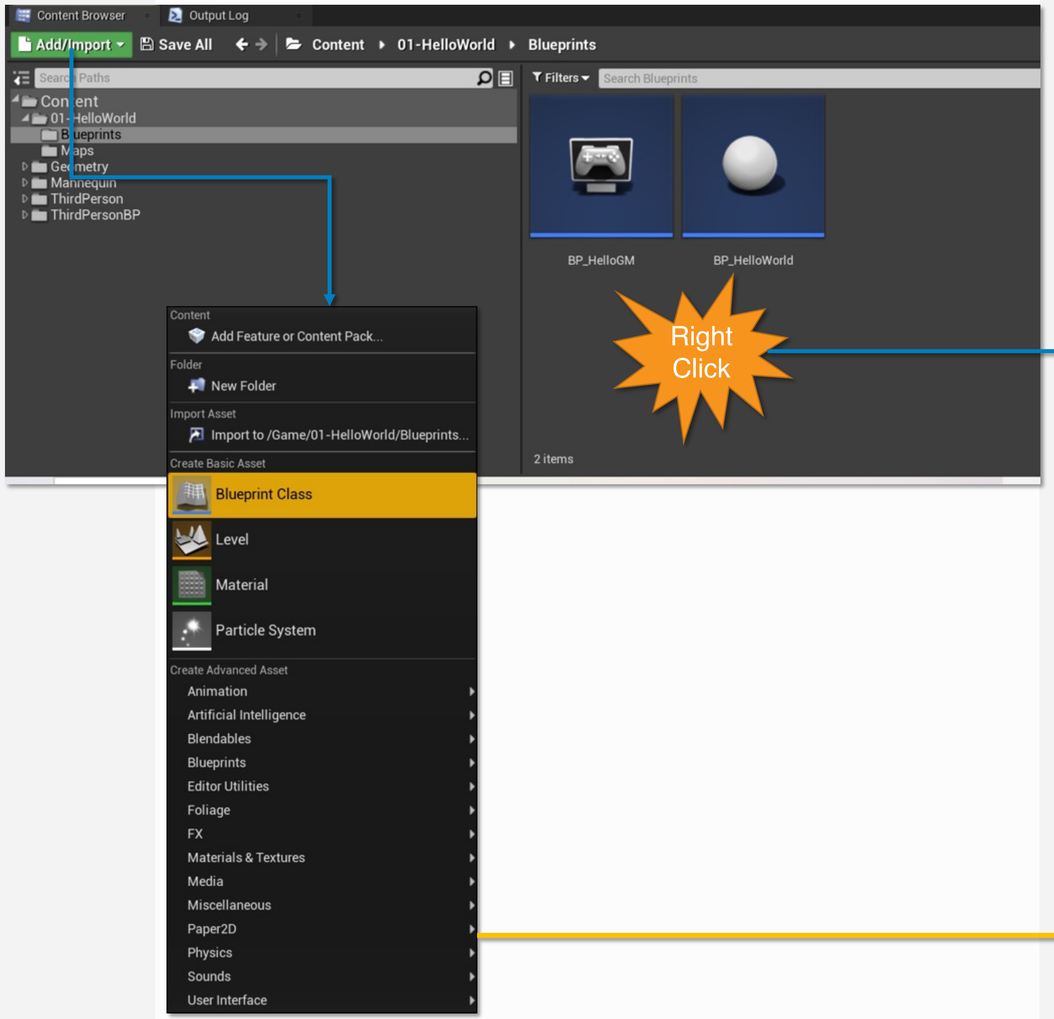
- 难道要编程了吗?
- 是的！我们要编程了！

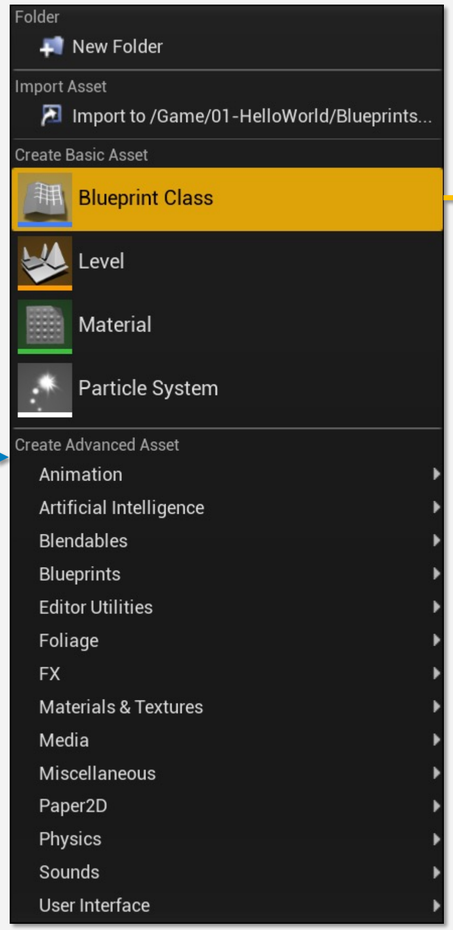


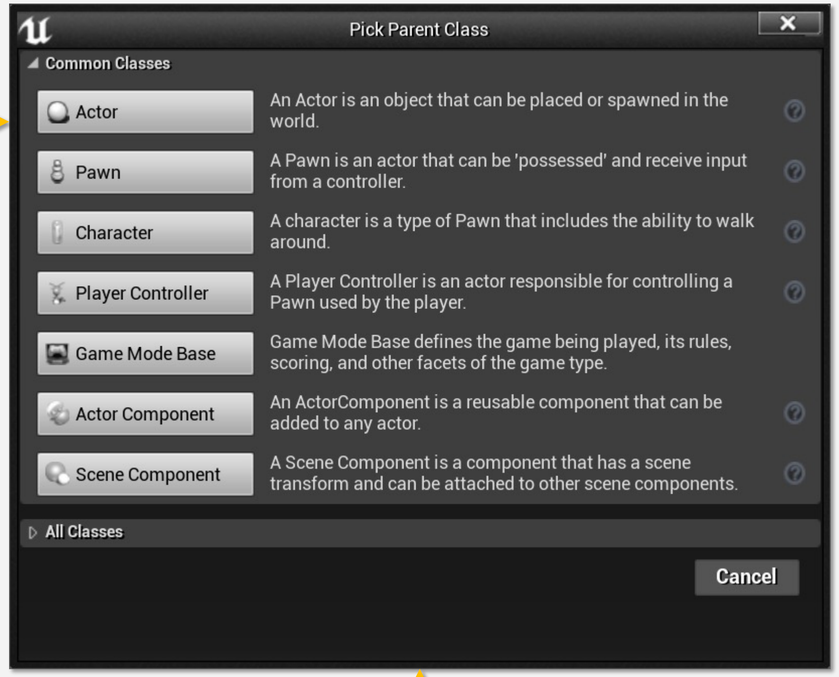
创建蓝图类

Creating Blueprint Classes

从 Content Browser 新建







Object
Actor
Actor Component

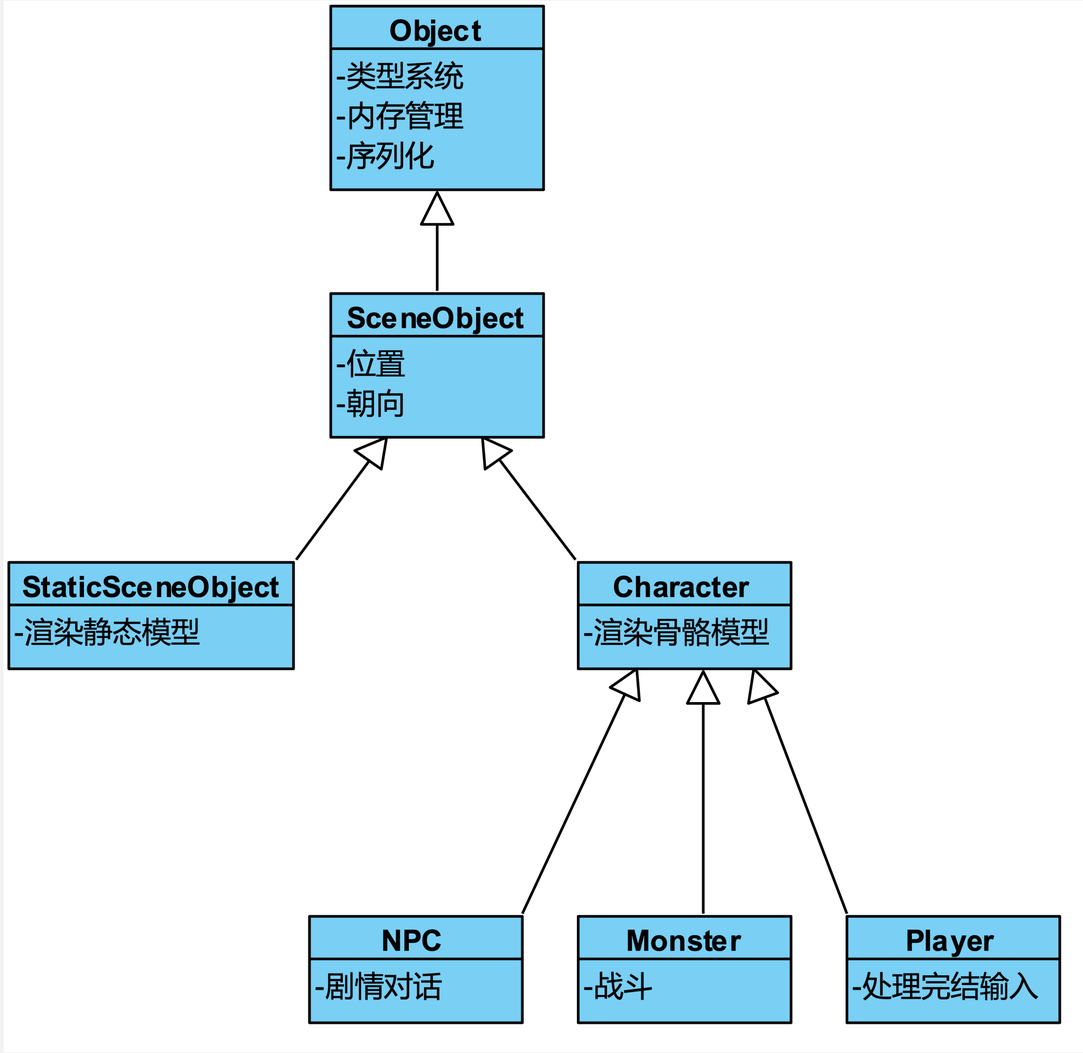
在创建新蓝图类时，必须定义**父类**。父类的所有变量和操作都将成为新类的一部分，新类称为**子类**（child class或subclass）。这个概念叫做**继承**。
下面是一些父类：

- Object：虚幻引擎中对象的基类。所有其他类都是Object类的子类。
- Actor：可以在关卡中放置或产生的对象所使用的基类。
- Actor Component：组件的基类，这些组件定义可添加到Actor的可复用行为。

因此，每个Actor都是Object，但并非所有Object都是Actor。例如，Actor Component是Object，但不是Actor。

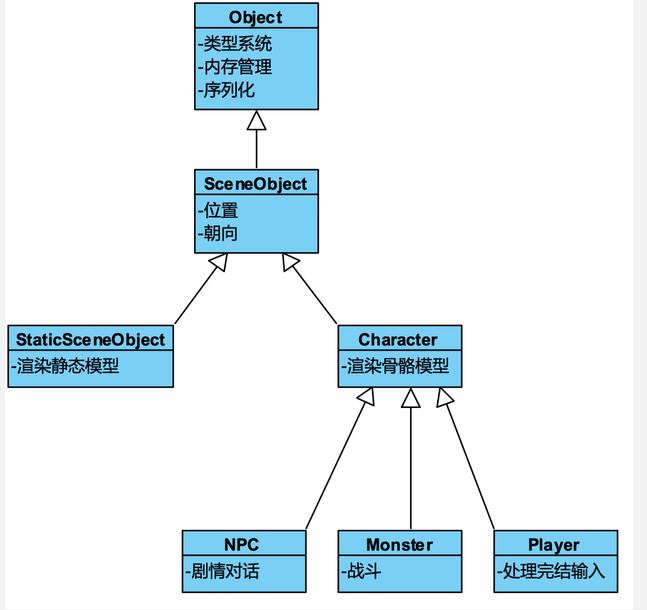
组件与现代面向对象的设计

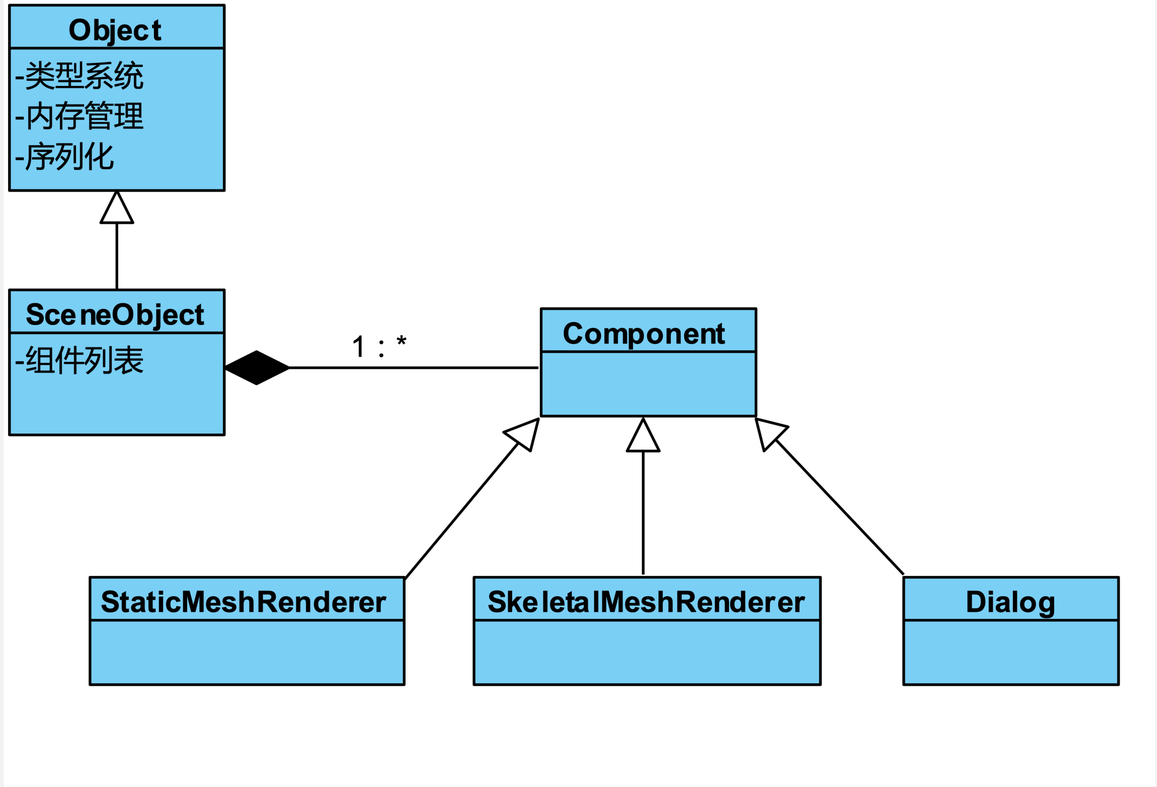
- 传统面对对象与现代面向对象
- 基于组件的对象设计在游戏引擎设计中的流行



组件与现代面向对象的设计

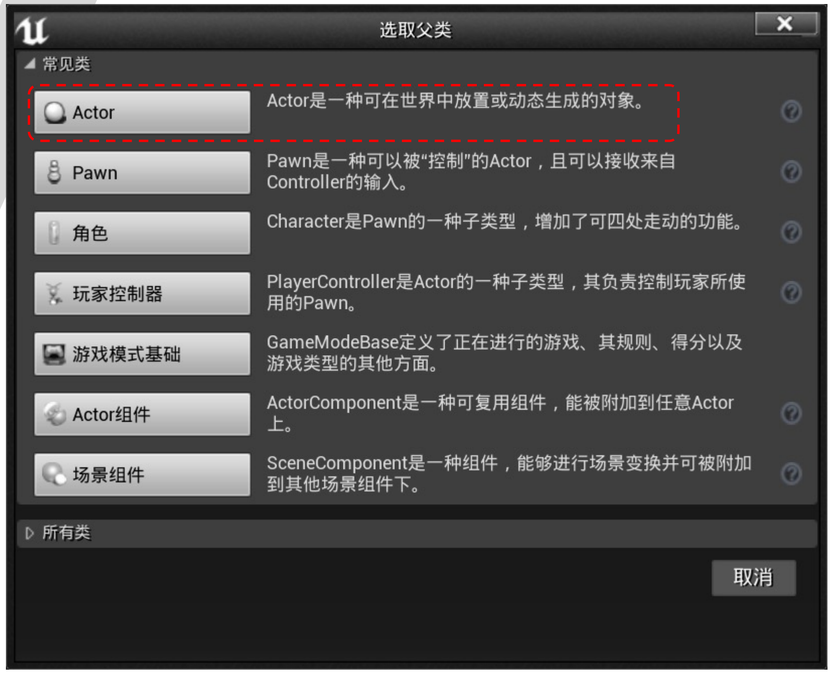
- 传统面对对象与现代面向对象
- 基于组件的对象设计在游戏引擎设计中的流行





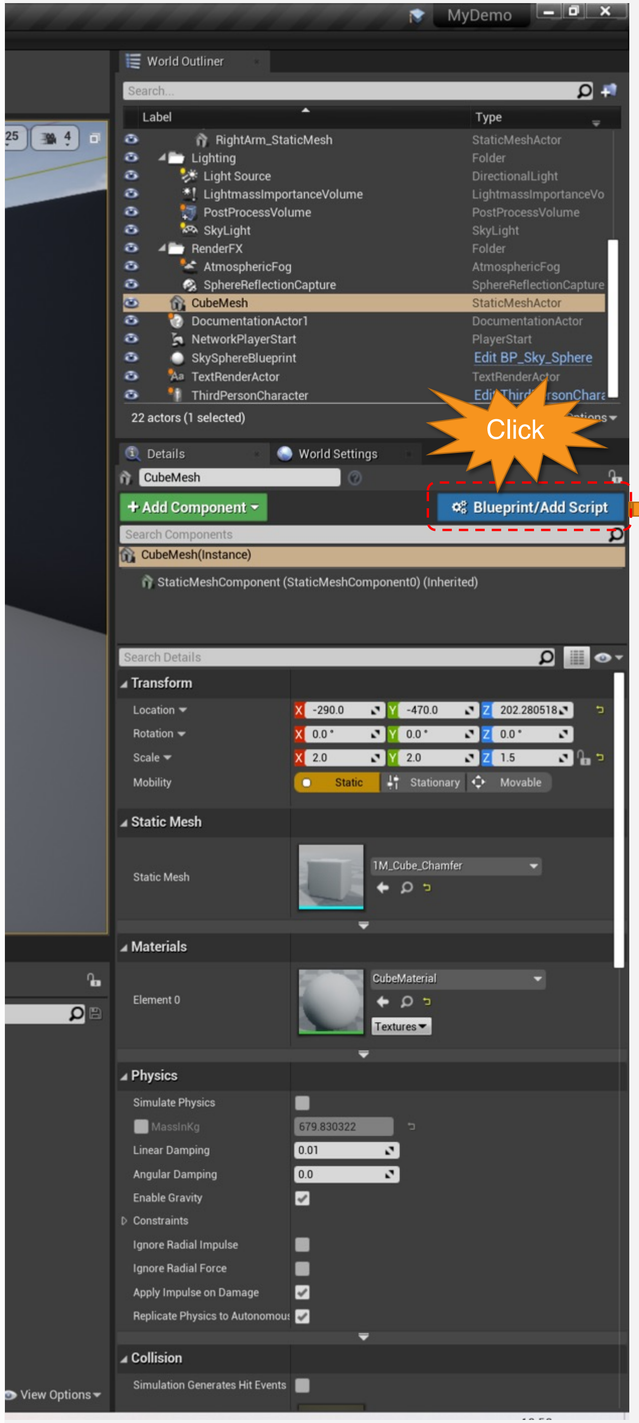
选取父类(Parent Class)

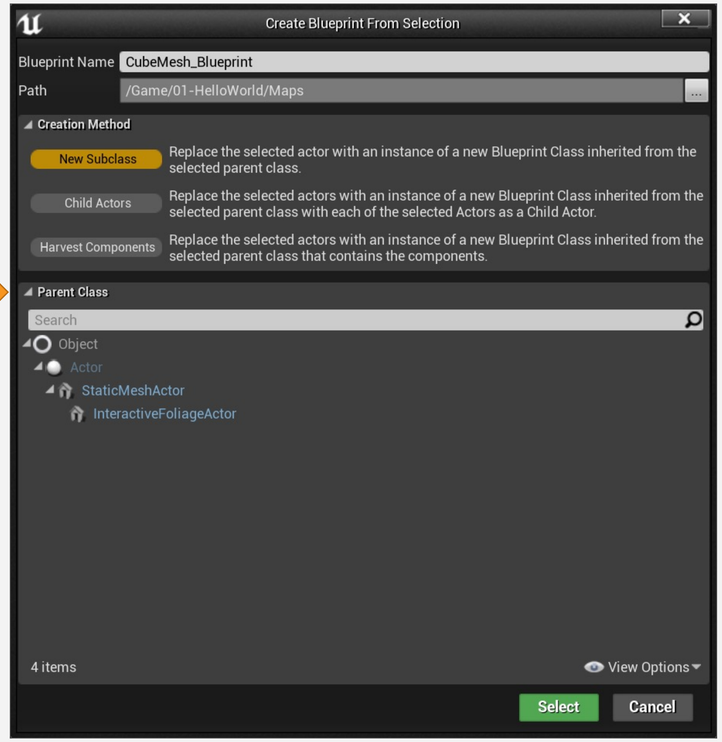
- 本周主要使用：**Actor**
- 下周详细讲解：OOD的典范



将关卡已有的Actor转换成蓝图

- 可能需要为关卡中的对象添加游戏逻辑
- 可能需要将简单对象替换成功能更强大的对象，并进行复用
- ……



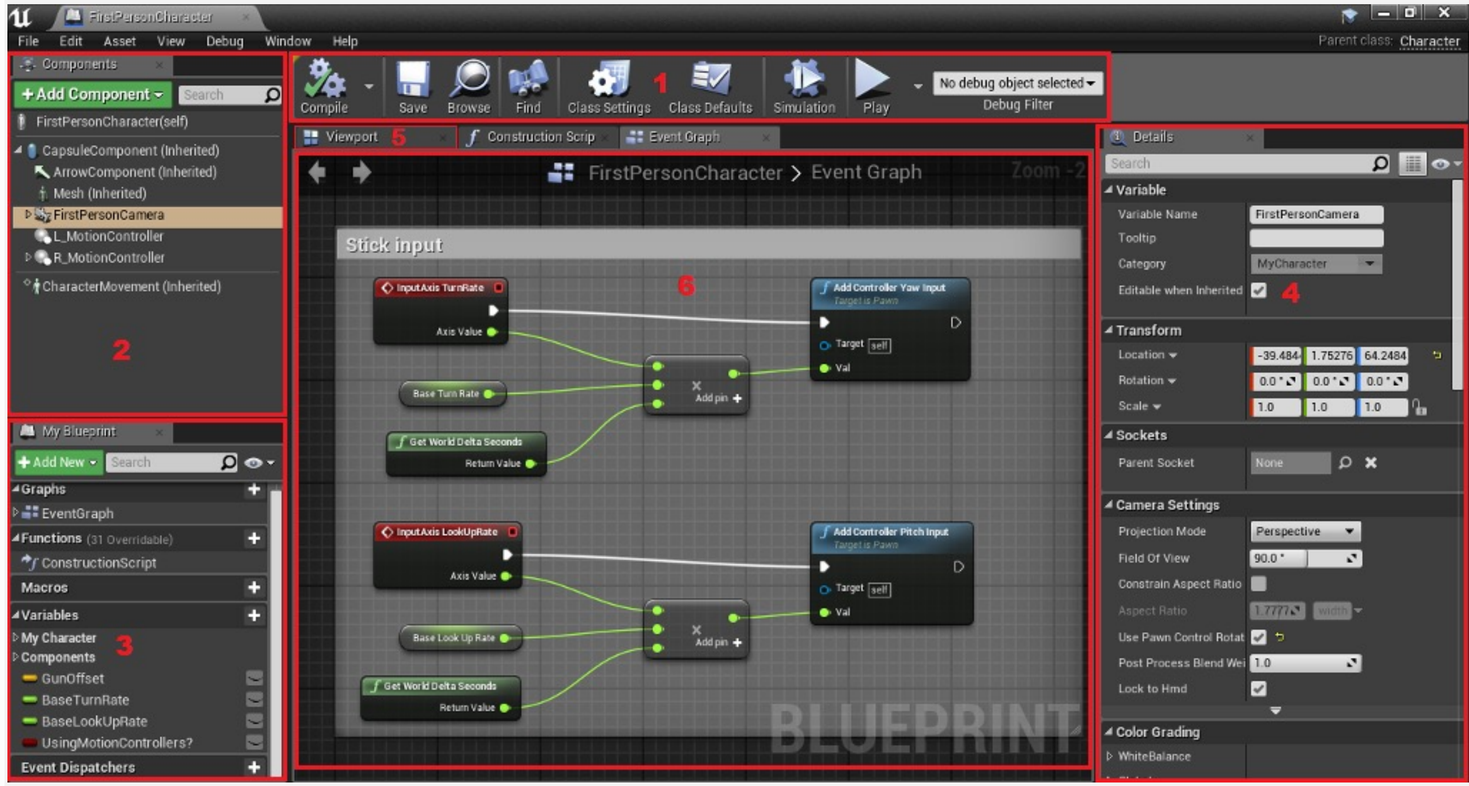


蓝图编辑器界面

Blueprint Editor Interface

蓝图编辑器概览

1. 工具栏
2. 组件（Components）面板
3. 我的蓝图（My Blueprint）面板
4. 细节（Details）面板
5. 视口（Viewport）
6. 事件图表（Event Graph）



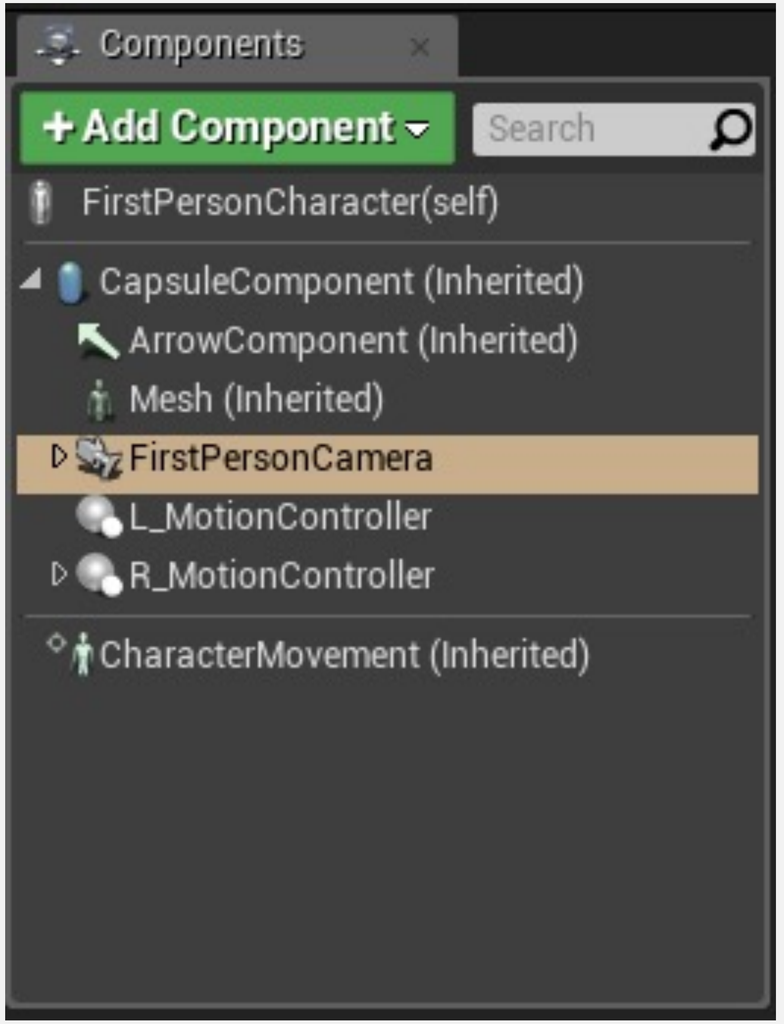
工具栏

编辑器顶部的工具栏包含一些基本蓝图编辑按钮：

- “**编译**”（Compile）：“编译”蓝图，这是验证节点和应用修改的必要操作
- “**保存**”（Save）：将对当前蓝图的所有更改保存到磁盘
- “**浏览**”（Browse）：在内容浏览器中显示当前蓝图
- “**查找**”（Find）：在蓝图中搜索节点
- “**类设置**”（Class Settings）：打开蓝图属性
- “**类默认值**”（Class Defaults）：允许修改蓝图变量的初始值

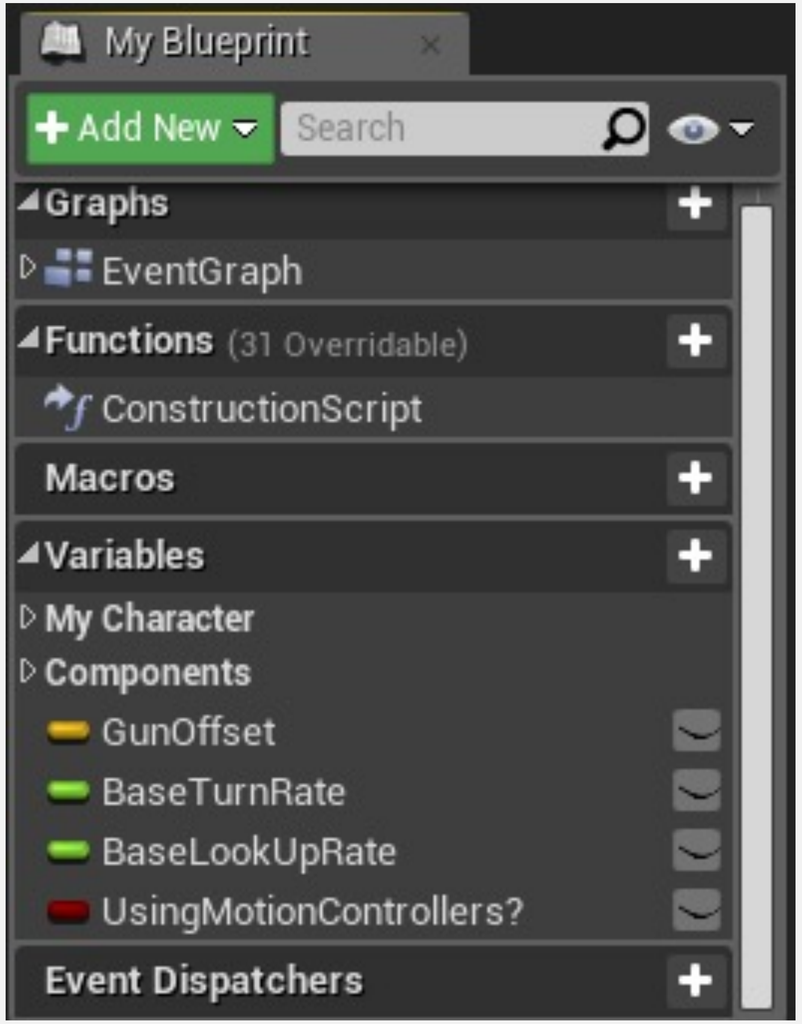
组件（Components）面板

- 在“组件”（Components）面板中，可以向当前蓝图添加各种类型的组件
- 常用组件包括静态网格体、光源、声音和碰撞测试中使用的几何体积



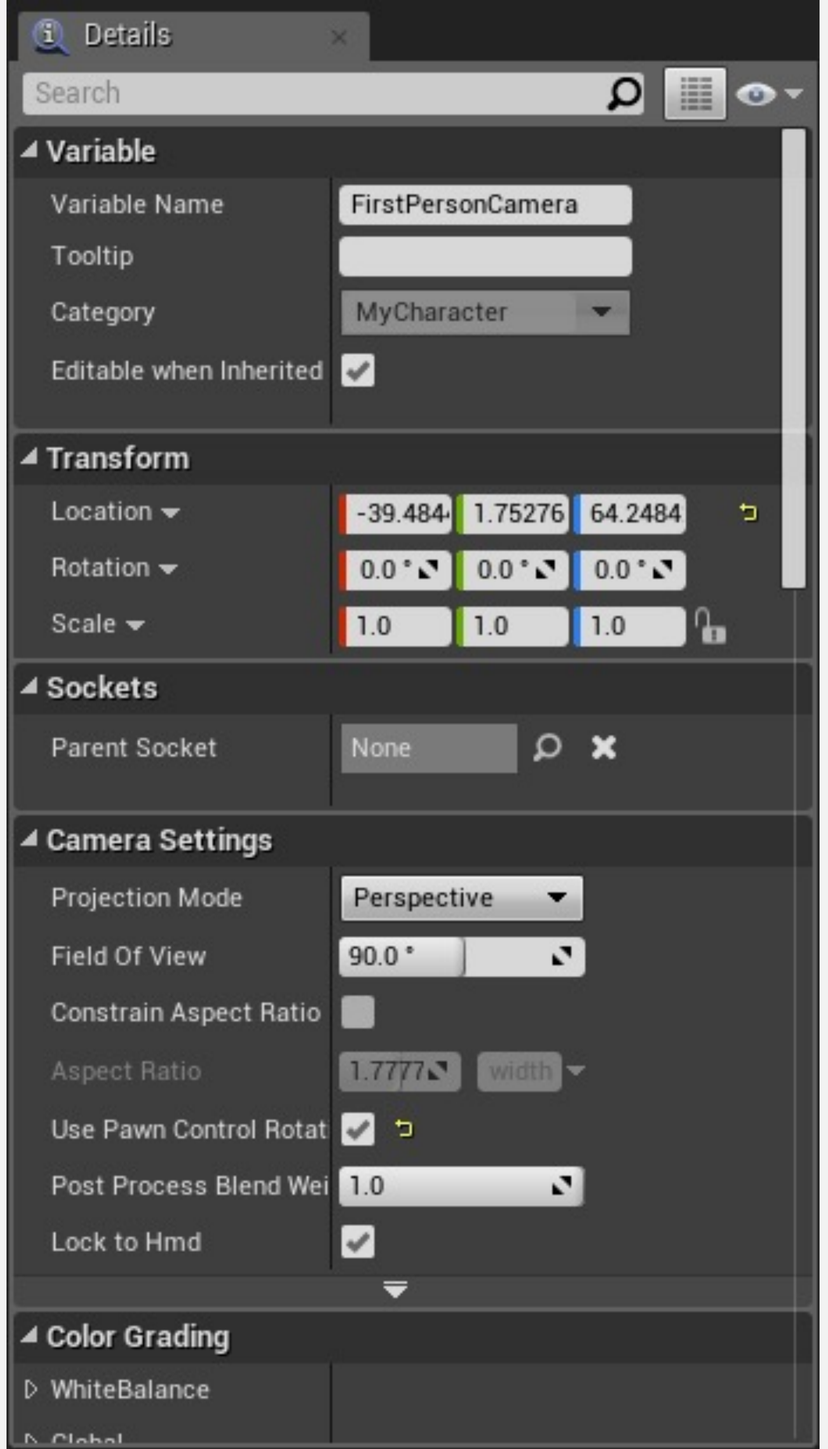
我的蓝图（My Blueprint）

- “我的蓝图”（My Blueprint）面板用于管理蓝图的变量、宏、函数和图表
- 它划分成多个类别，每个类别有一个“+”按钮，用于添加新元素



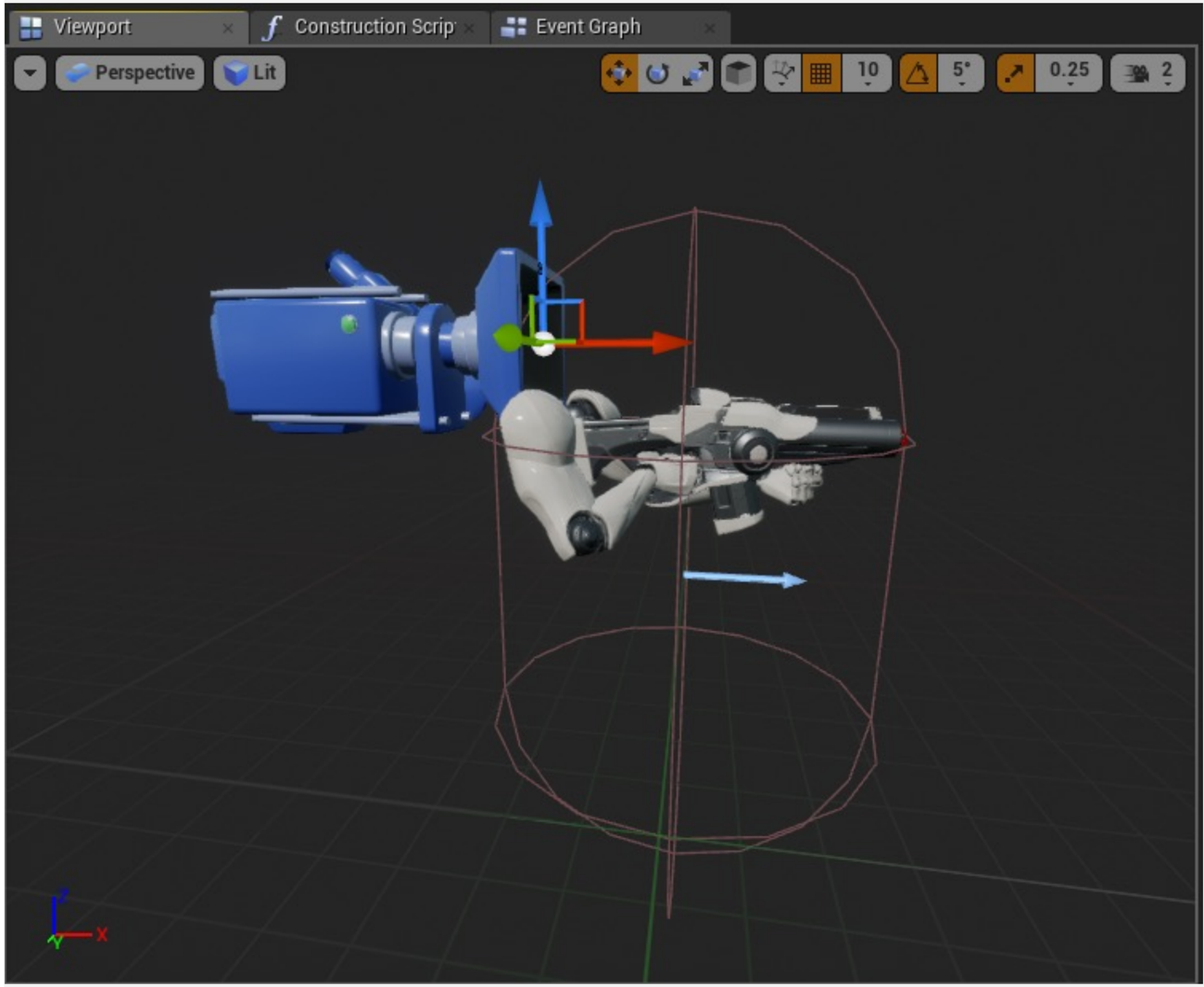
细节（Details）面板

- “细节”（Details）面板显示蓝图类当前选中元素（可以是变量、函数或组件）的属性。
- 这些属性按类别归类，它们的值可以修改。
- 面板顶部有一个搜索框，可以用于过滤属性。



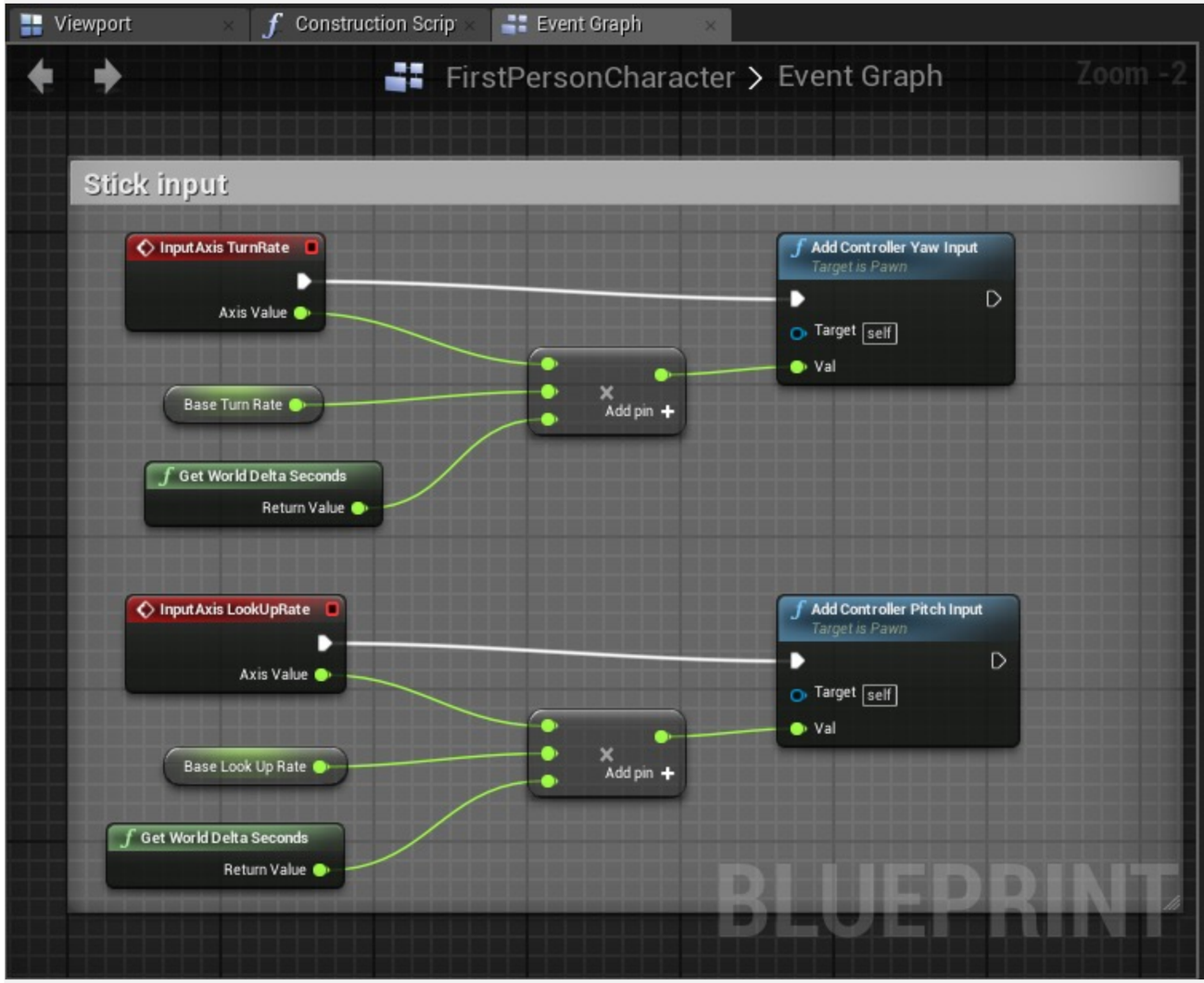
视口（Viewport）

- 视口包含属于蓝图一部分的组件的视觉表示
- 可以使用变换（Transform）控件在视口中对组件进行操控，就像在关卡编辑器中一样



事件图表（Event Graph）

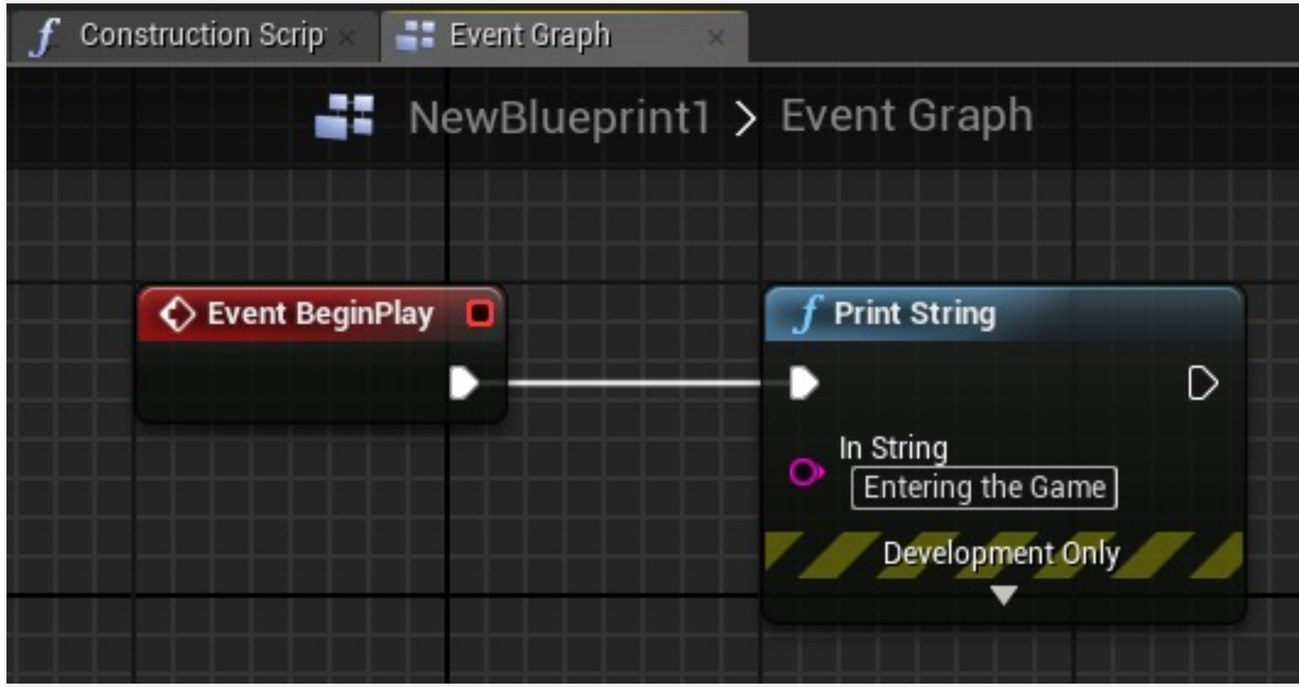
- 事件图表是 Gameplay 逻辑，用于确定游戏运行期间蓝图类的行为
- 事件图表包含节点图表所表示的事件和操作



BeginPlay 事件

事件让虚幻引擎和Actor进行通信。常见示例是开始播放（BeginPlay）事件。

当Actor的游戏开始时，将触发开始播放（BeginPlay）事件。如果在游戏中产生了Actor，则该事件会立即触发。



Tick 事件

有一个事件会在游戏每一帧调用，称作“Tick”事件。例如，在每秒60帧的速度运行的游戏中，一秒会调用60次Tick事件。

Tick事件有一个参数叫做“变化秒数”（Delta Seconds），它包含自上一帧以来经过的时间。

在右图所示的Tick事件中，Actor沿着X轴以每秒100厘米的速度移动。

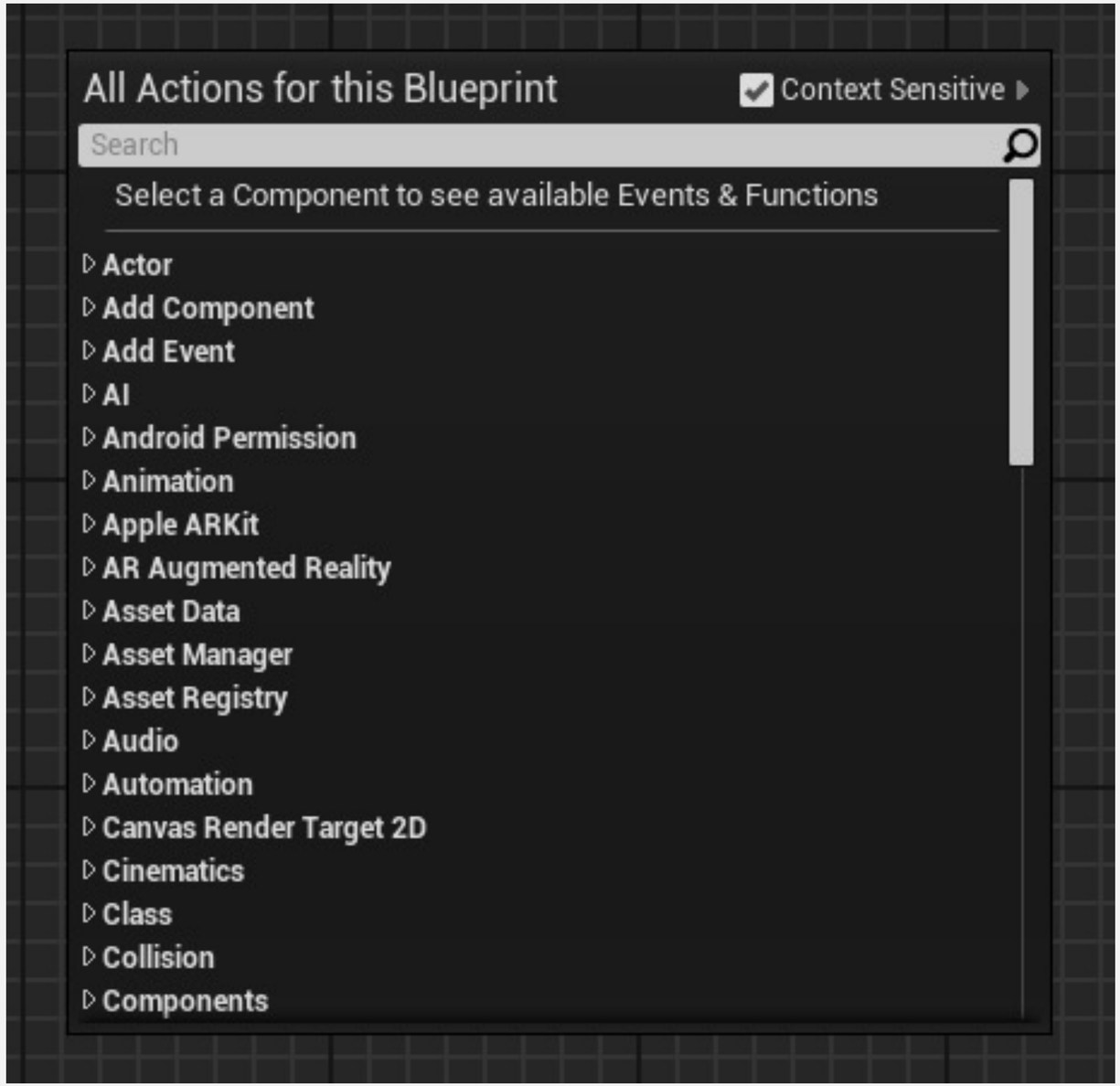
Tick事件应当仅在必要时使用，因为它会影响性能。

放置节点

Placing Nodes

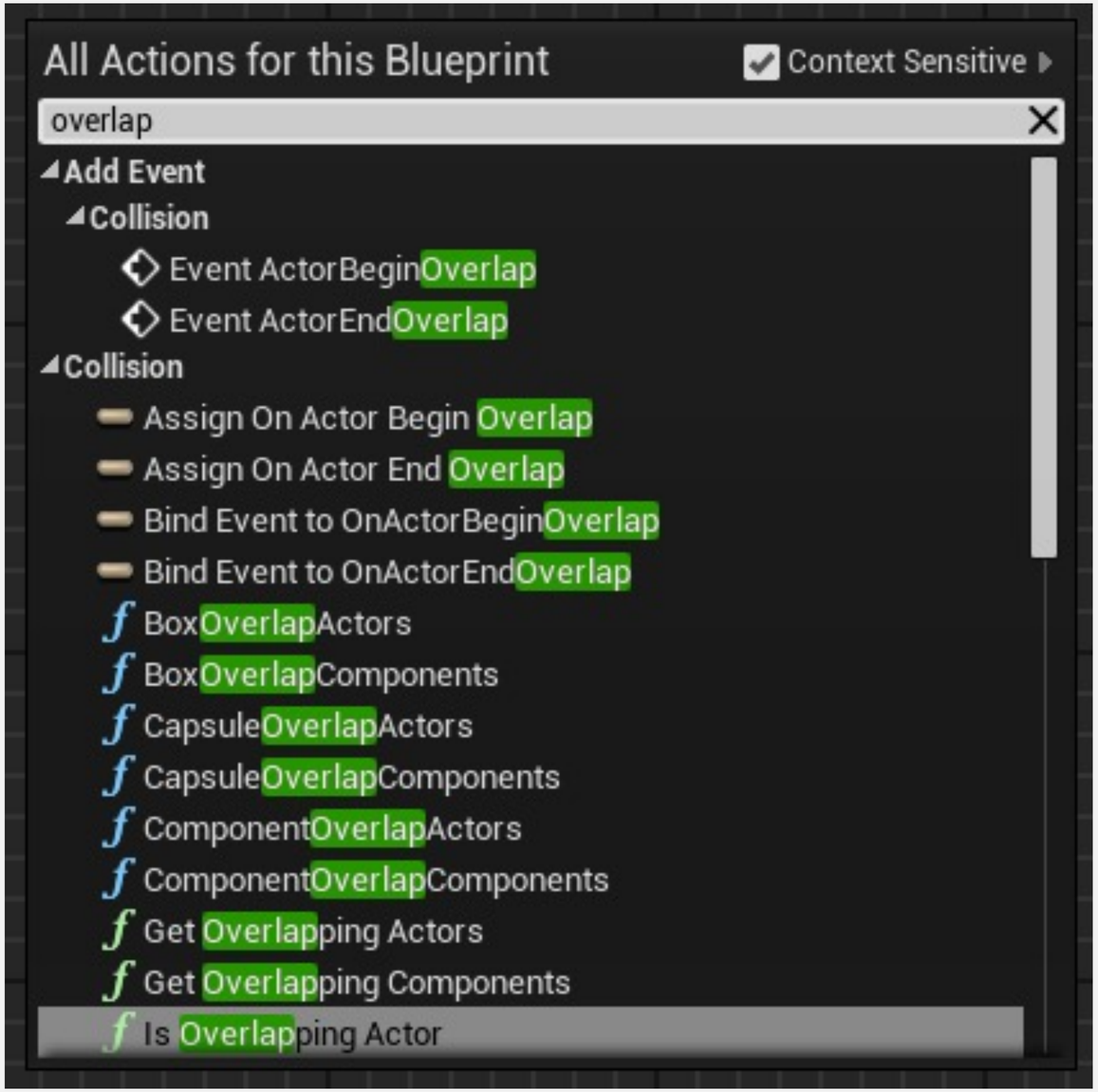
快捷菜单

- 快捷菜单用于向图表添加节点
- 节点表示变量、运算符、函数调用和事件
- 打开快捷菜单：在事件图表的空位置**单击右键**，或者从节点的**引脚拖出引线**，然后松开鼠标按钮



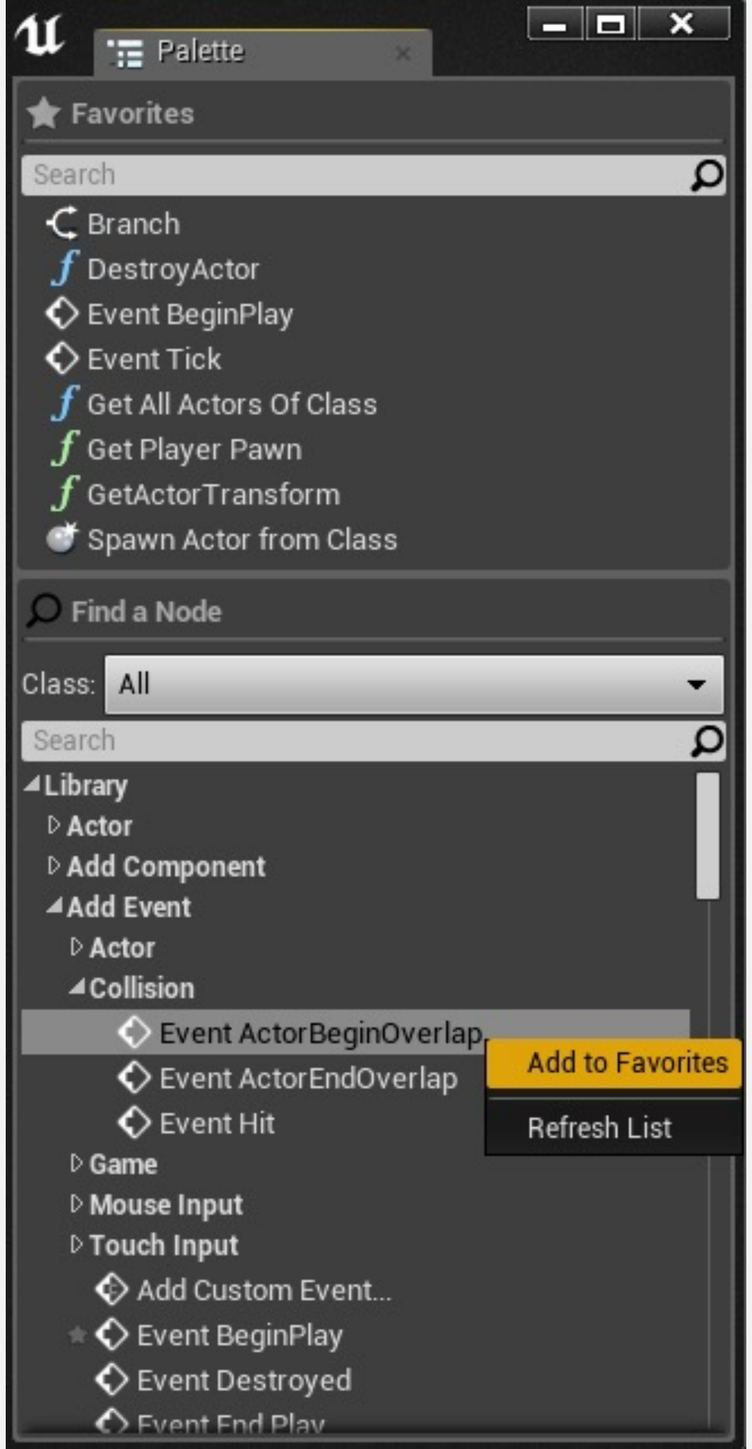
搜索栏

- 快捷菜单包含一个搜索栏，用于过滤用户所在进行输入时显示的节点列表
- “控制板”（Palette）面板有两个搜索栏，一个用于“常用”（Favorites）部分，一个用于“控制板”（Palette）列表



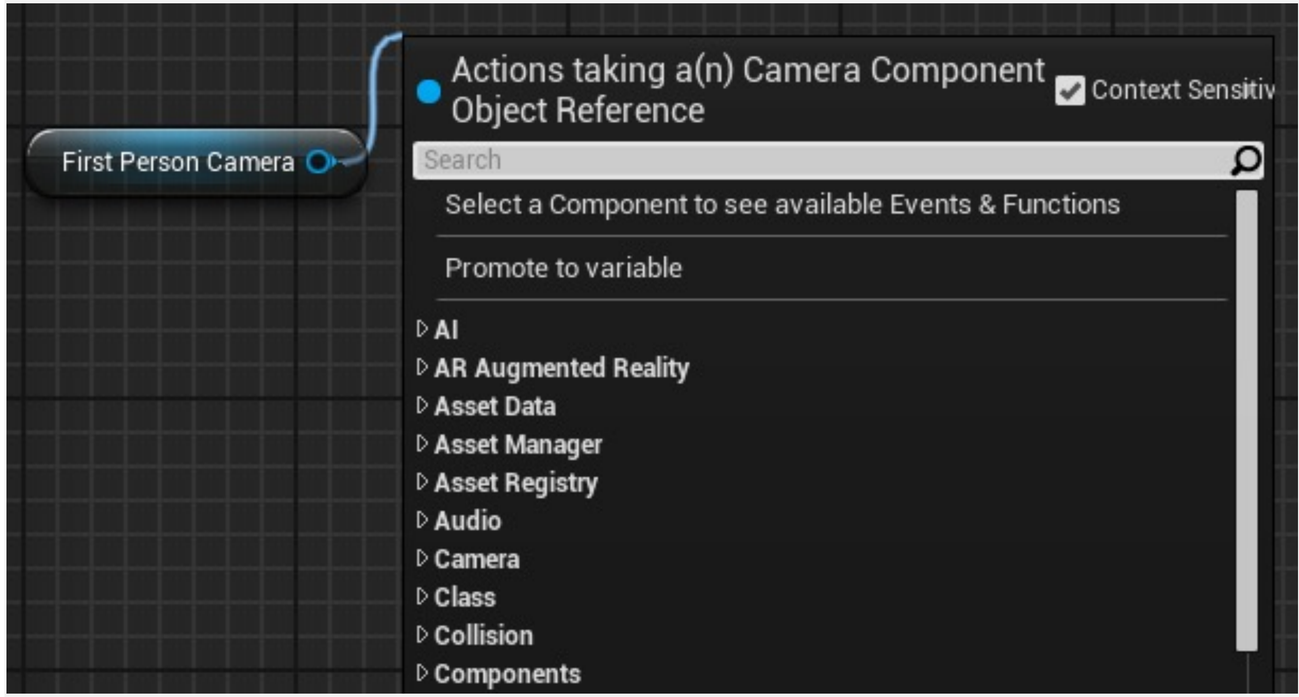
控制板（Palette）面板

- 在蓝图编辑中，前往“窗口”（Window）>“控制板”（Palette）可以打开“控制板”（Palette）面板
- “控制板”（Palette）面板包含所有可以在蓝图中使用的节点列表
- 该面板顶部是“常用”（Favorites）部分，显示您喜欢的最常用节点
- 要将节点设置为常用节点，在“控制板”（Palette）面板右键单击这个节点，然后选择“添加到常用”（Add to Favorites）



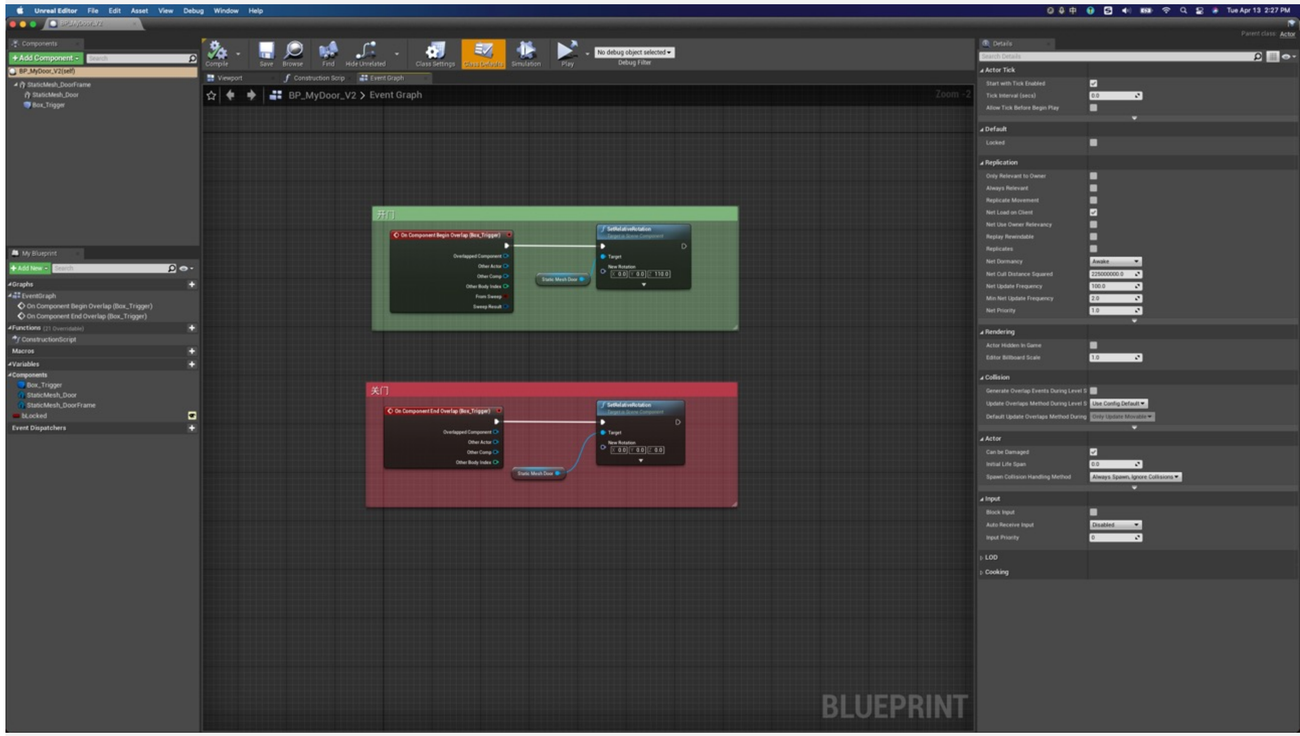
情境关联(Context Sensitive)

- 在快捷菜单顶部，有一个复选框，名为“情境关联”（Context Sensitive）
- 如果选中该框，则节点列表会根据当前情境中可以使用的操作进行筛选
- 如果快捷菜单是通过在事件图表中单击右键而打开的，则情境为当前蓝图类
- 如果是通过从节点引脚拖出引线而打开的，则情境为该引脚类型



Event Graph 其他操作

- 添加注释
- 节点单选 & 多选
- 节点删除
- 连线&断开
- 窗口缩放



小目标1

自动开启的门

小目标1

玩家靠近之后，自动开启的门：

- 使用 **Static Mesh 组件**渲染门的模型
- 使用 **Box Collision 组件**触发事件
- 动画过程：下周使用 Timeline 完成

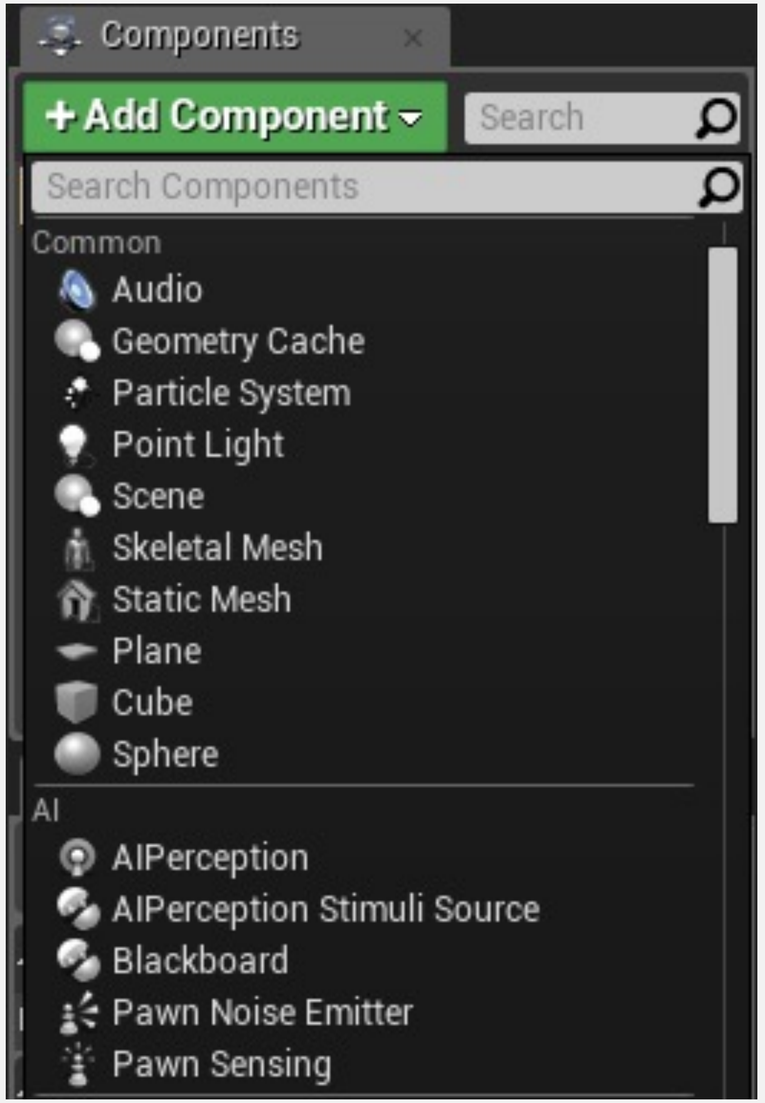
构建Actor类

Building Actor Classes

组件（Components）

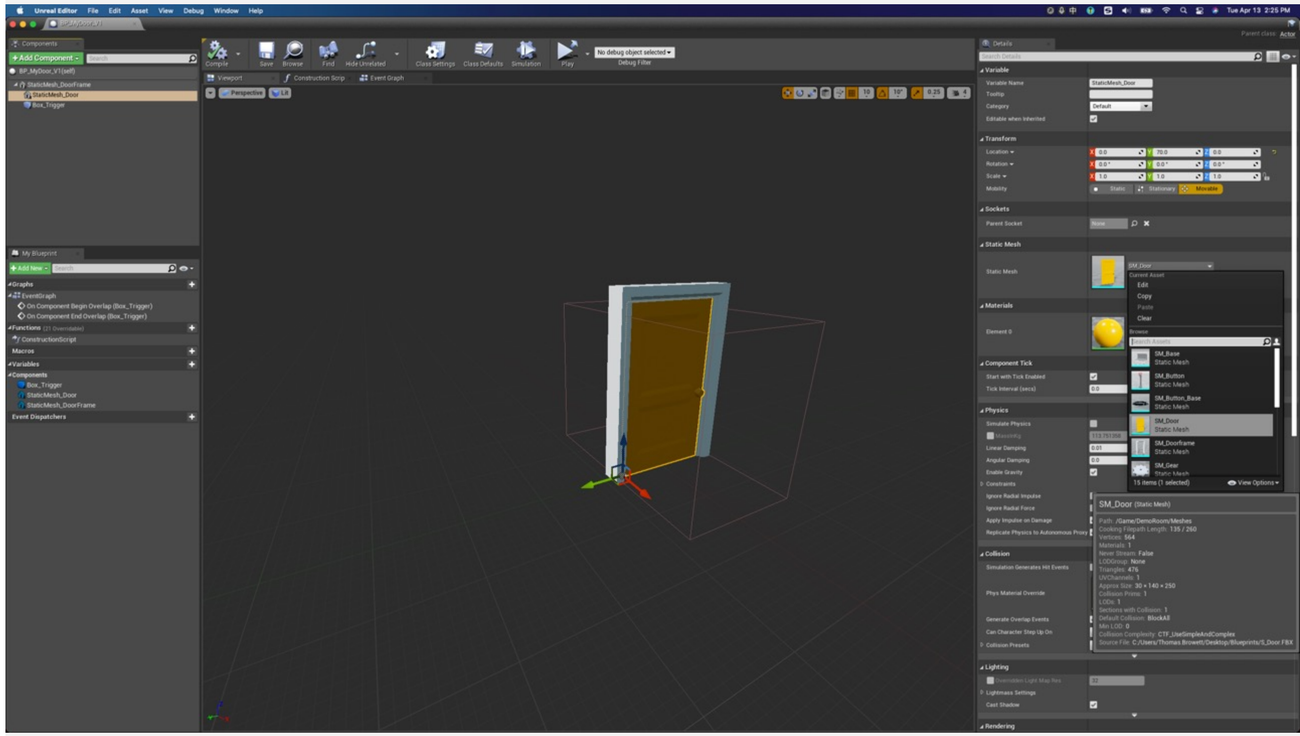
- 组件是可复用的功能模块，可以组合到蓝图类中
- 引擎提供了 40+ 可复用的组件
- 使用蓝图编辑器中的“组件”（Components）面板。

右图显示了一个蓝图的“组件”（Components）面板，点击“添加组件”（Add Component）按钮后显示的下拉菜单选项。



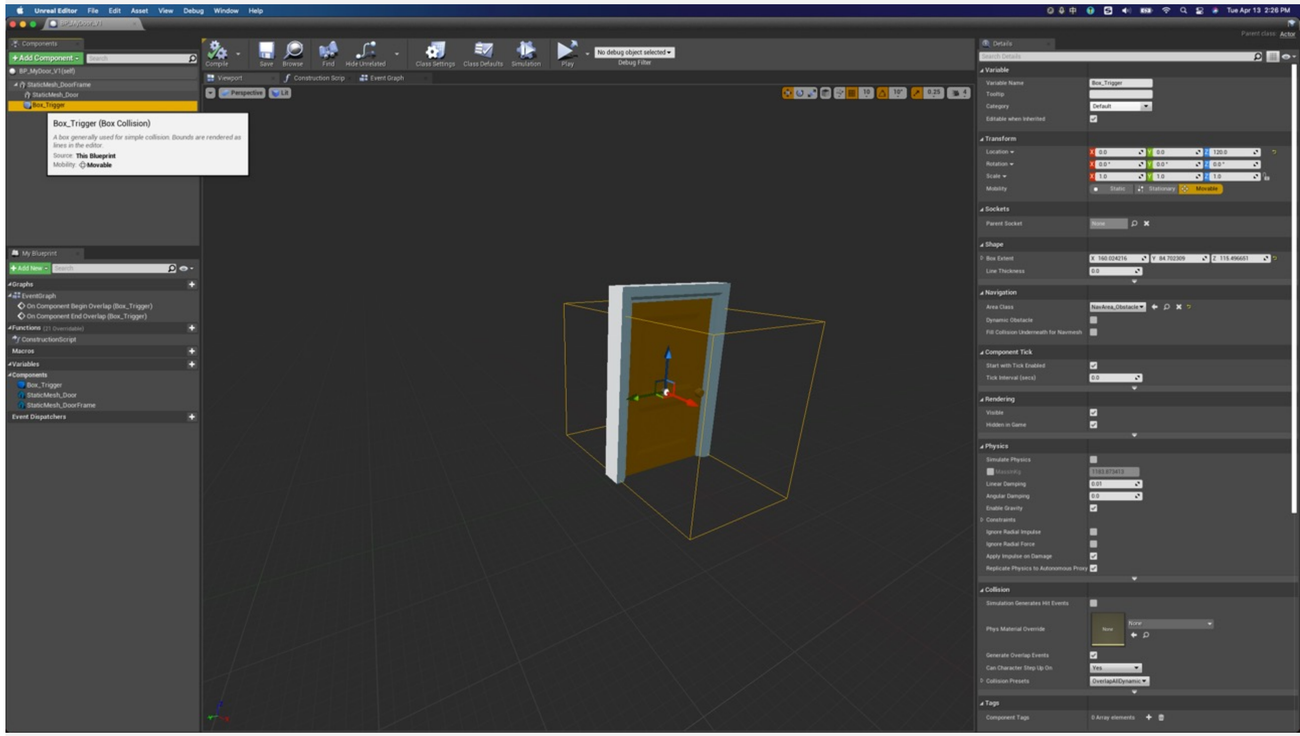
Static Mesh 组件

- 用来渲染指定的静态模型
- 可以控制相对位置、朝向等属性值



Box Collision 组件

- 引擎提供了多个碰撞体组件，Box、Sphere 开销最小
- 可以触发碰撞相关事件

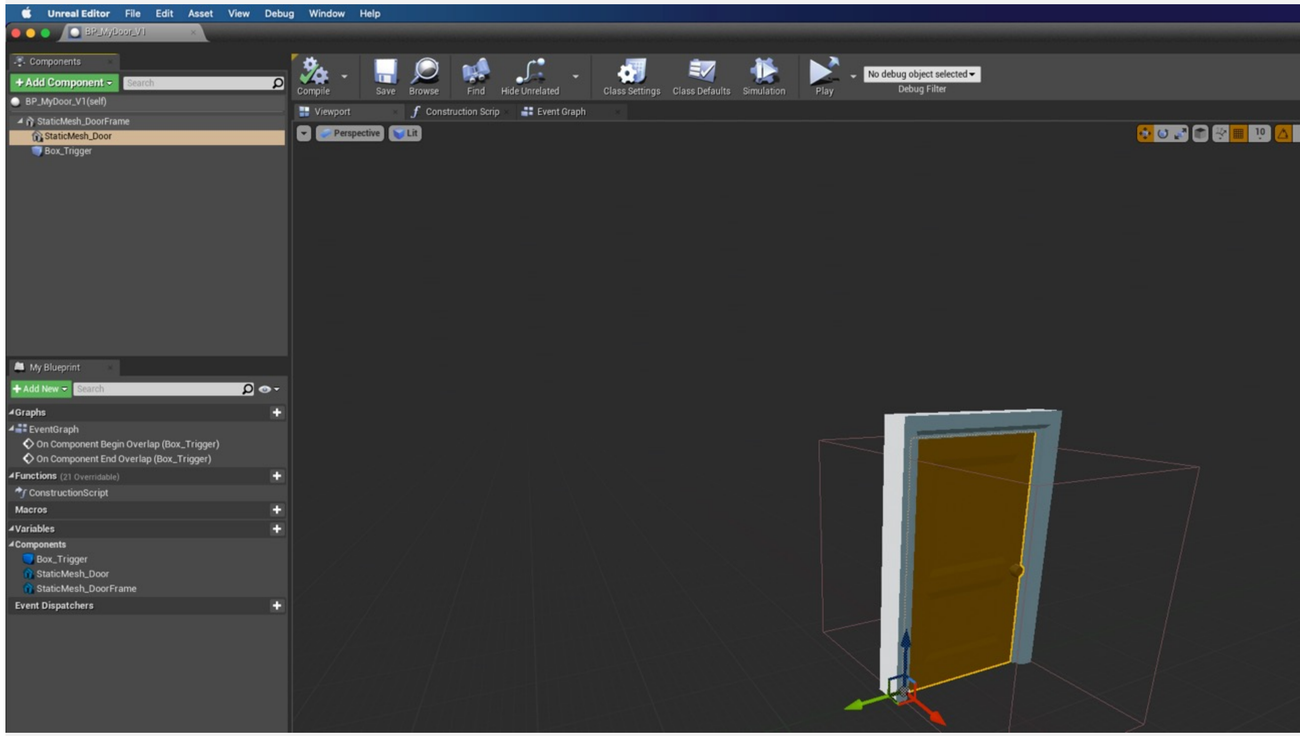


组件与视口

- Actor 的视觉表现可以在视口中预览
- 被选中的组件会在视口中高亮显示，其属性可以通过 Details 面板编辑

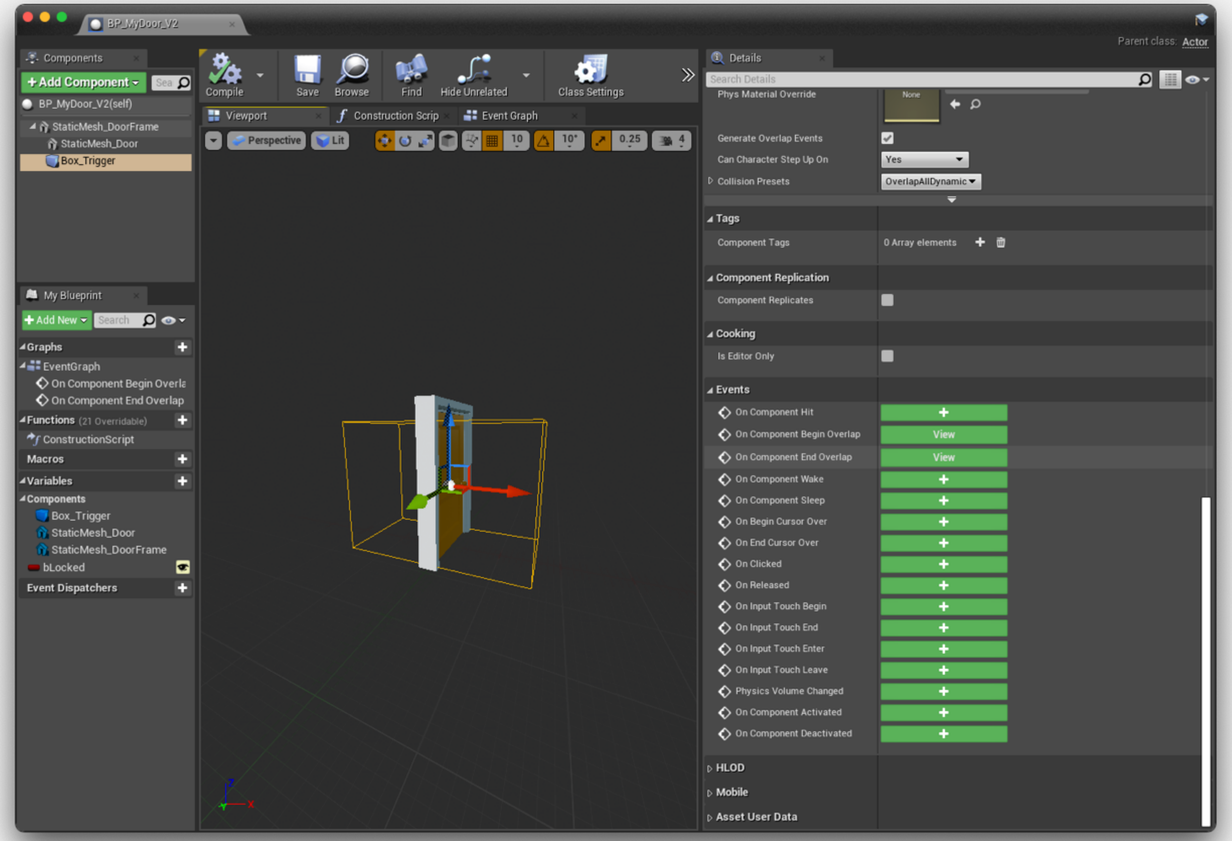
右图的这个蓝图类显示包含：

- 两个 Static Mesh 组件
- 一个 Box Collision 组件



组件的事件响应

- 被选中的组件，其事件会在 Details 面板中显示
- 点击 “+” 按键可以新建事件响应节点
- 点击 “View” 查看已有的事件响应节点



小目标2

锁住的门

小目标2

如果门是锁住的，则不会自动开启：

- 使用一个**变量**控制门是否上锁
- 使用**分支(Branch)**节点处理“上锁”的逻辑

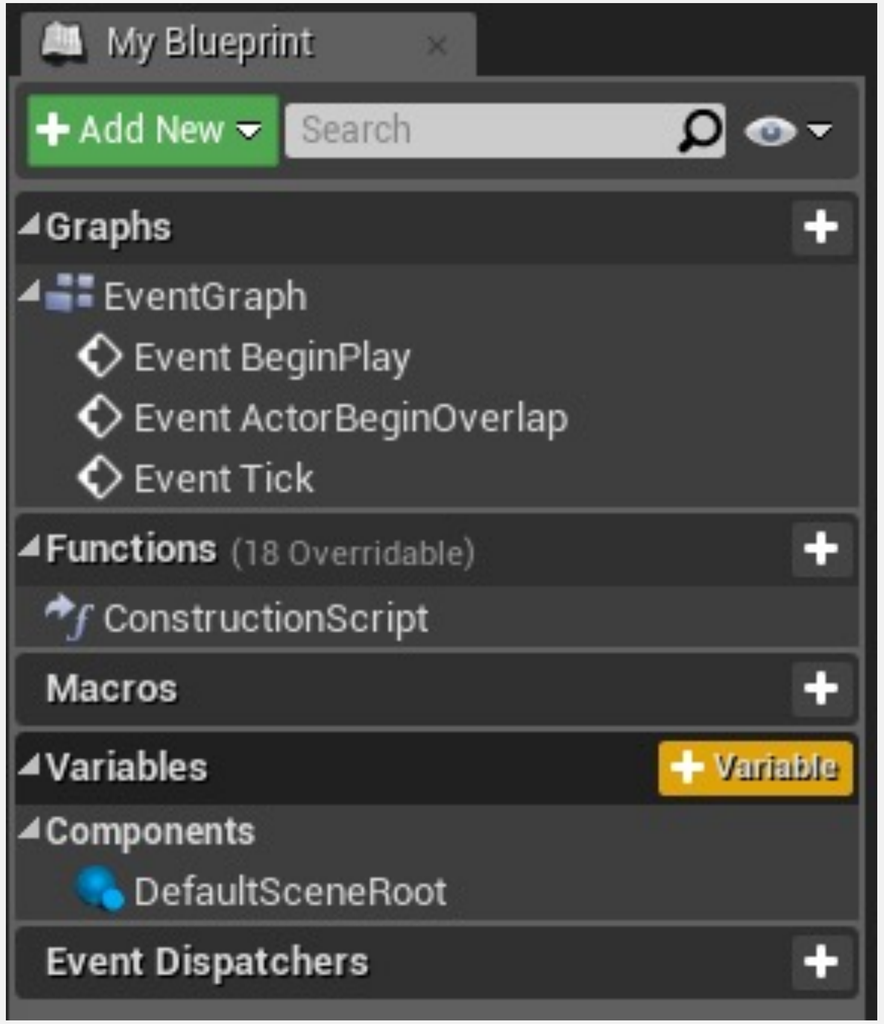
变量

Variables

创建变量

变量用于存储蓝图中的值和属性，它们可以在游戏执行期间修改。变量可以有不同的类型。

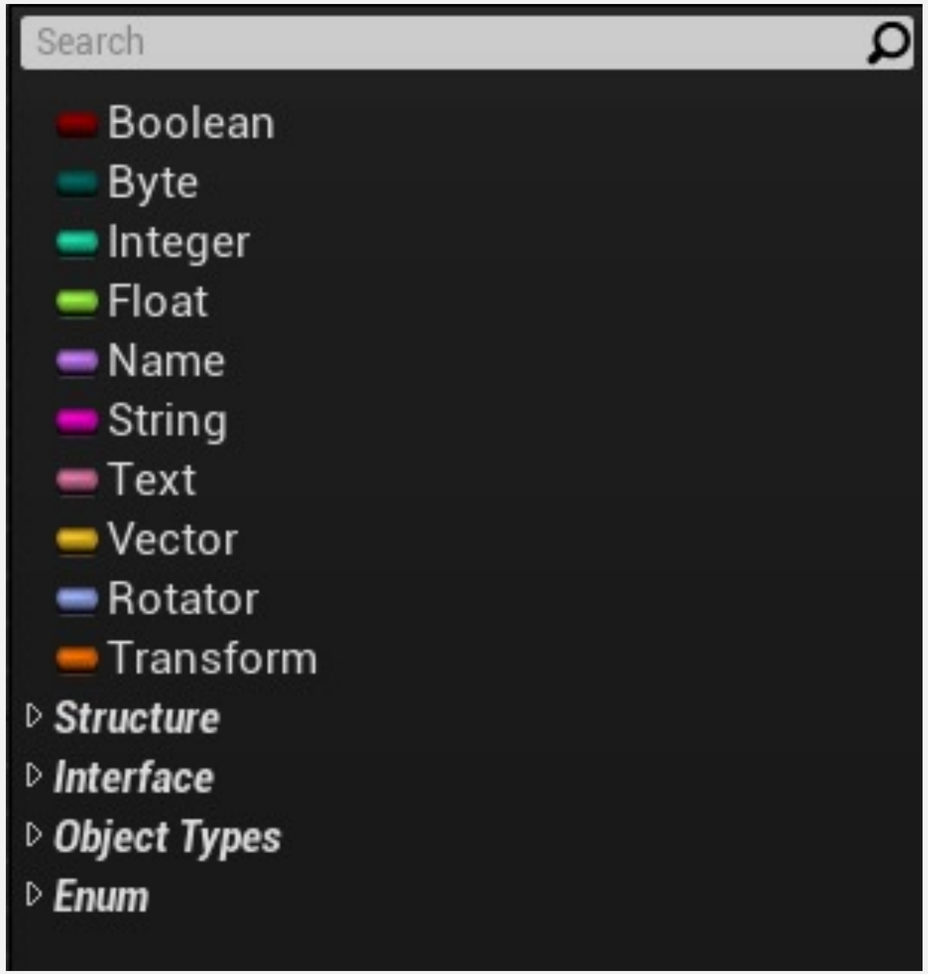
要创建变量，前往蓝图编辑器中的“我的蓝图”（My Blueprint）面板，单击“变量”（Variables）类别中的“+”按钮。



变量数据类型

蓝图是一种静态类型（Static Typed）的脚本语言，以下是一些常见的变量类型：

- **布尔**（Boolean）：只能保存值“true”或“false”
- **整数**（Integer）：用于存储整数值
- **浮点数**（Float）：用于存储小数值
- **字符串/文本**（String / Text）：用于存储文本。首选文本变量，因为它支持本地化
- **矢量**（Vector）：包含浮点值X、Y和Z
- **变换**（Transform）：用于存储位置、旋转和缩放。

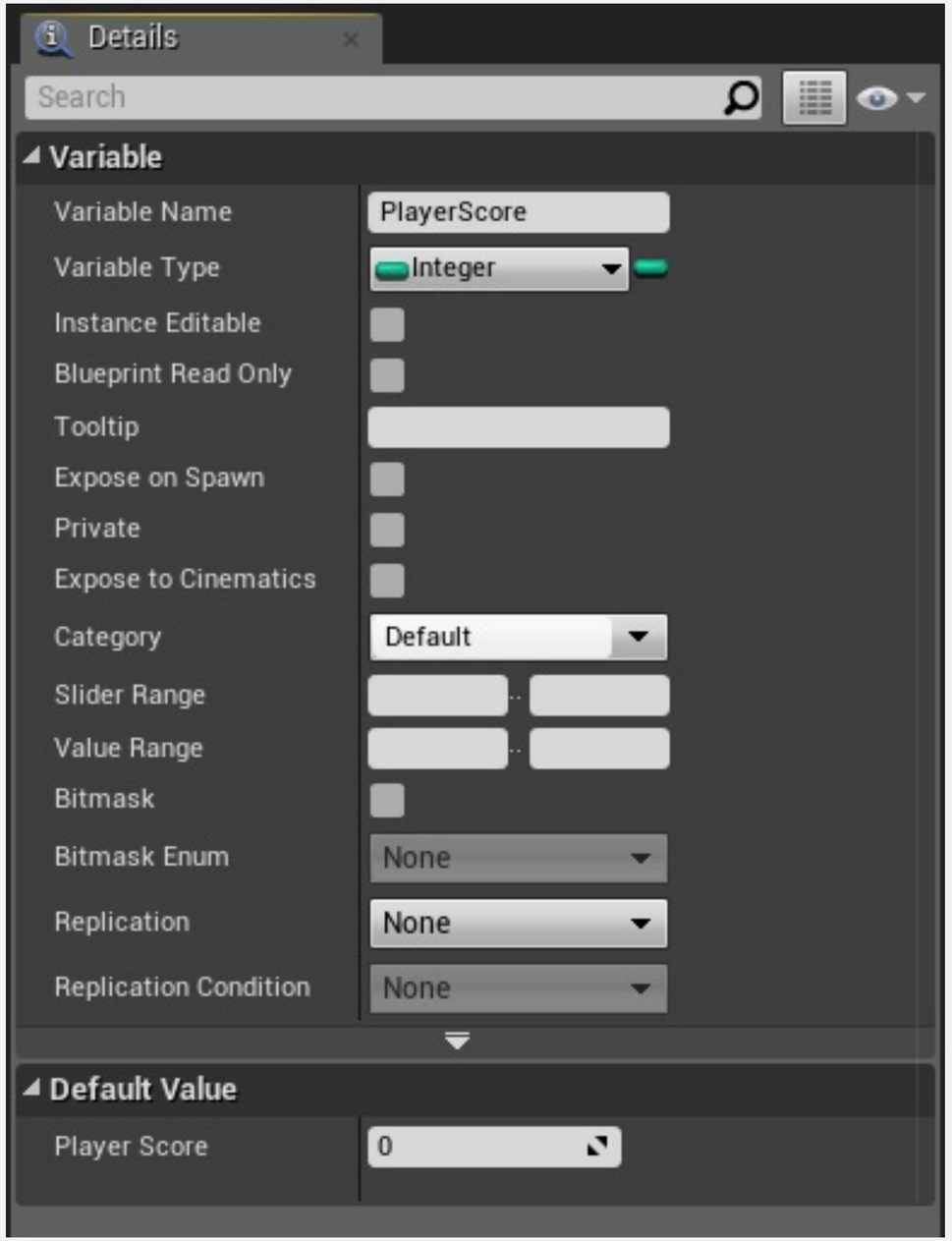


变量的细节（Details）面板

选择变量时，其属性显示在“细节”（Details）面板中。在这个面板中，可以更改变量名称和类型。

“细节”（Details）面板中的其他属性包括：

- 实例可编辑（Instance Editable）：如果选中，则可以在关卡中的实例中更改变量
- 只读蓝图（Blueprint Read Only）：如果选中，则蓝图节点不能更改变量
- 工具提示（Tooltip）：包含光标悬浮于变量上时显示的信息
- 在生成时显示（Expose on Spawn）：如果选中，则可以在产生蓝图时设置变量

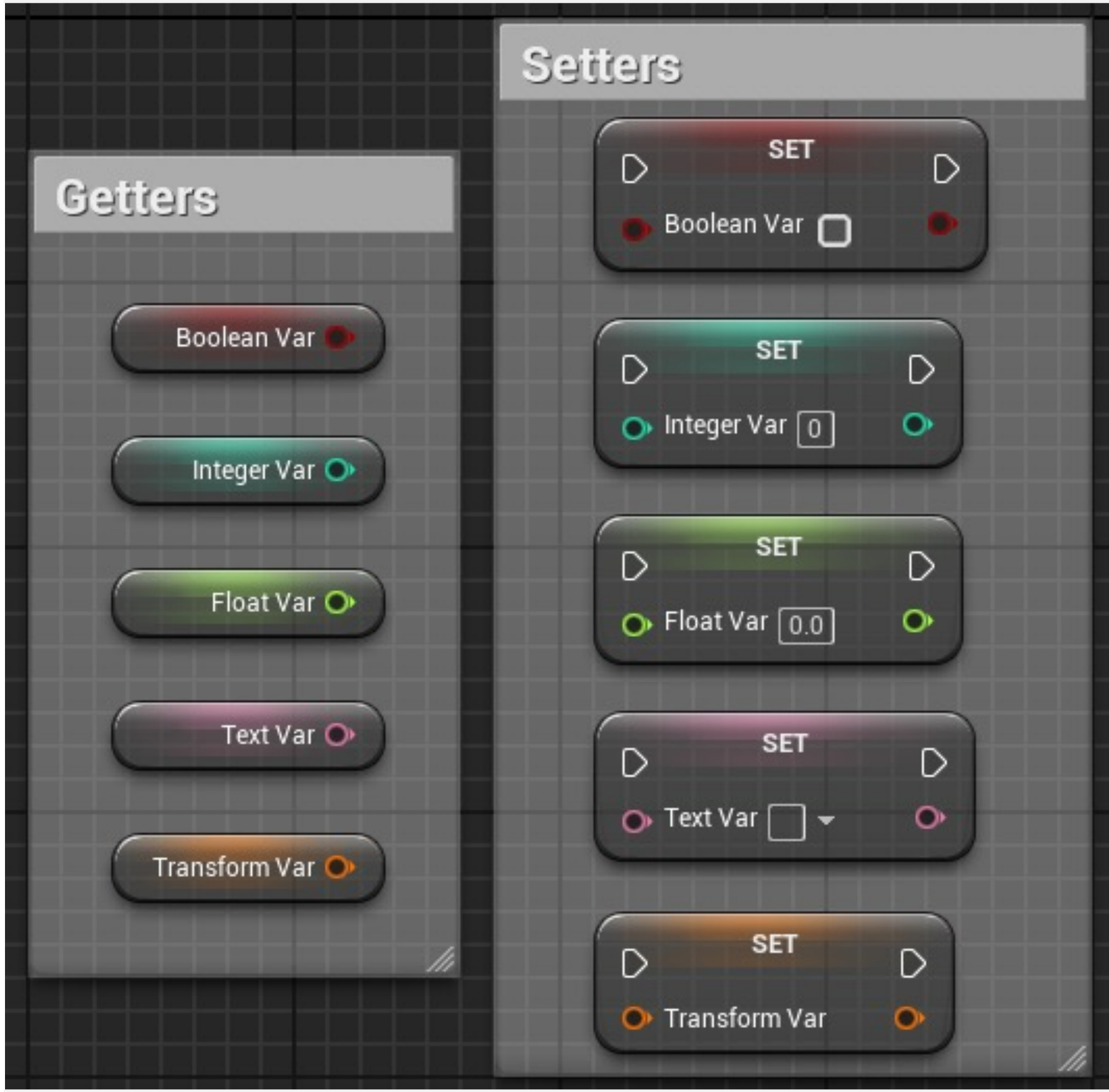


Getter 和 Setter

将变量拖放到事件图表中时，将出现一个包含Get和Set选项的快捷菜单。

- Get节点用于读取变量的值。
- Set节点用于在变量中存储新值。

有一些有用的快捷键可用来创建Get和Set节点。要创建Get节点，拖放变量时按Ctrl键。Set节点使用Alt键创建。

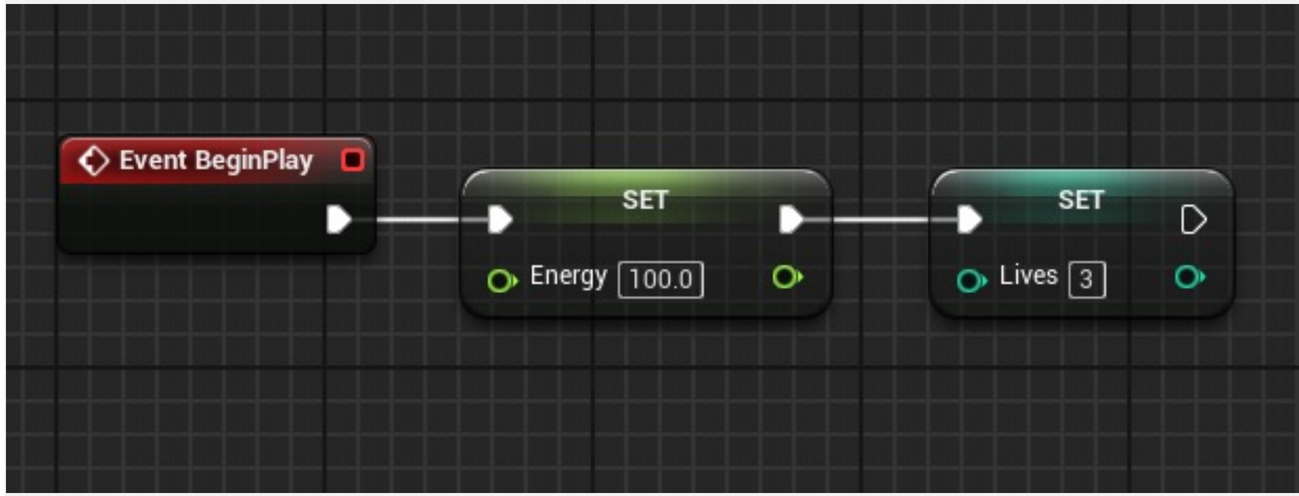


节点、引脚与引线

Nodes, Pins & Wires

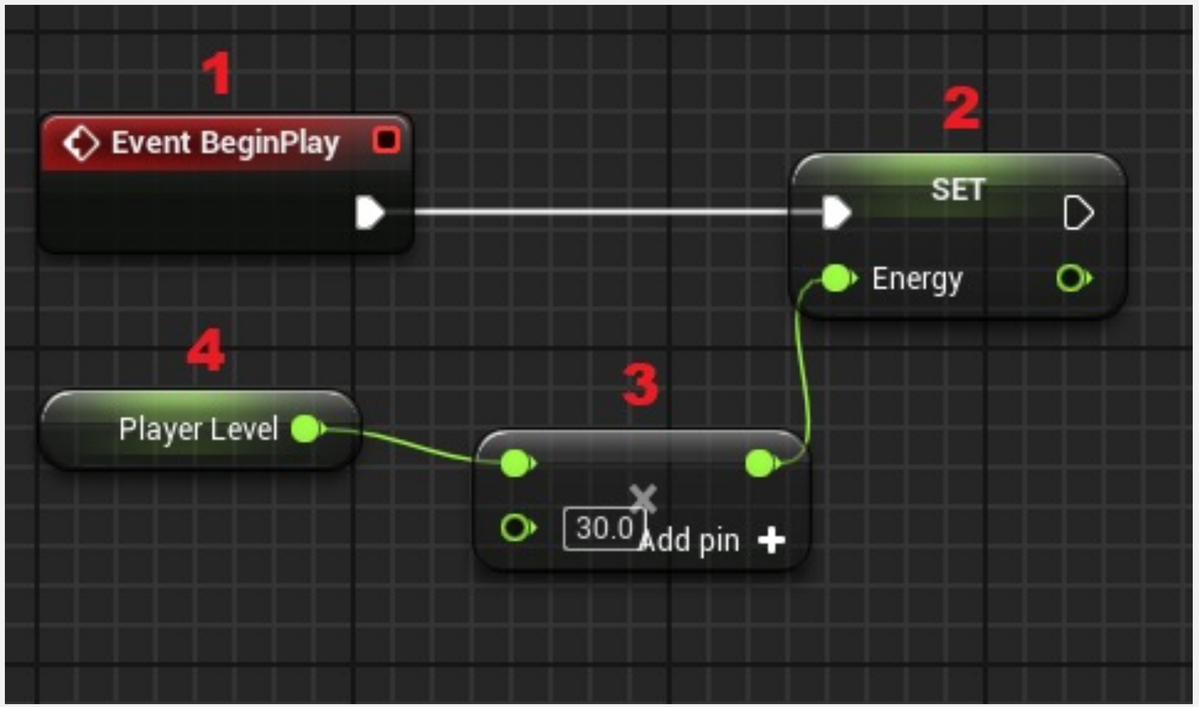
蓝图图表执行

- 蓝图是事件驱动的（event based）
- 蓝图节点执行始于红色事件节点，从左到右沿着白色引线执行，直到到达最后一个节点
- 节点的白色引脚称为执行引脚(execution pins)
- 其他有颜色的引脚称为数据引脚(data pins)



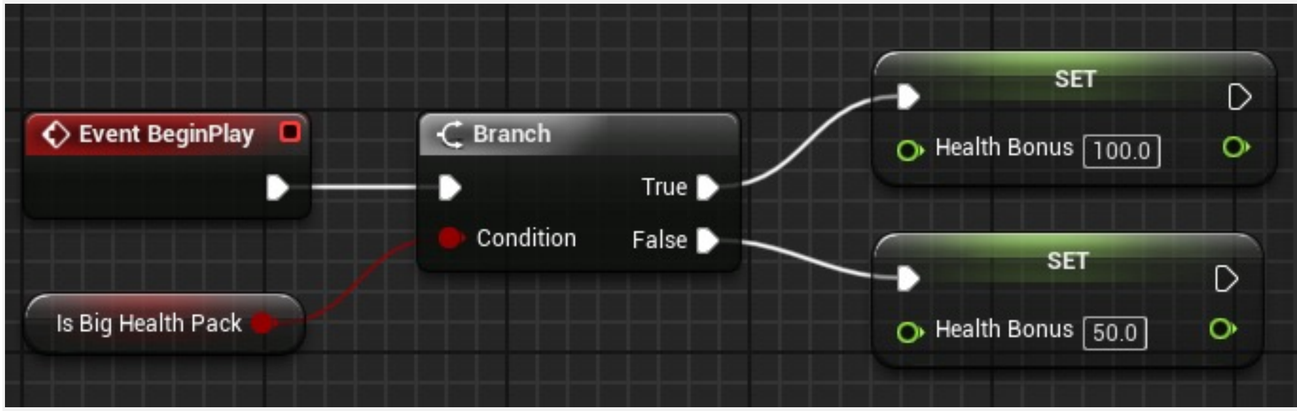
数据引线（Data Wires）

- 当具有数据引脚的节点运行时，它使用数据引线获取所需数据，然后再完成执行。
- 节点的输入&输出



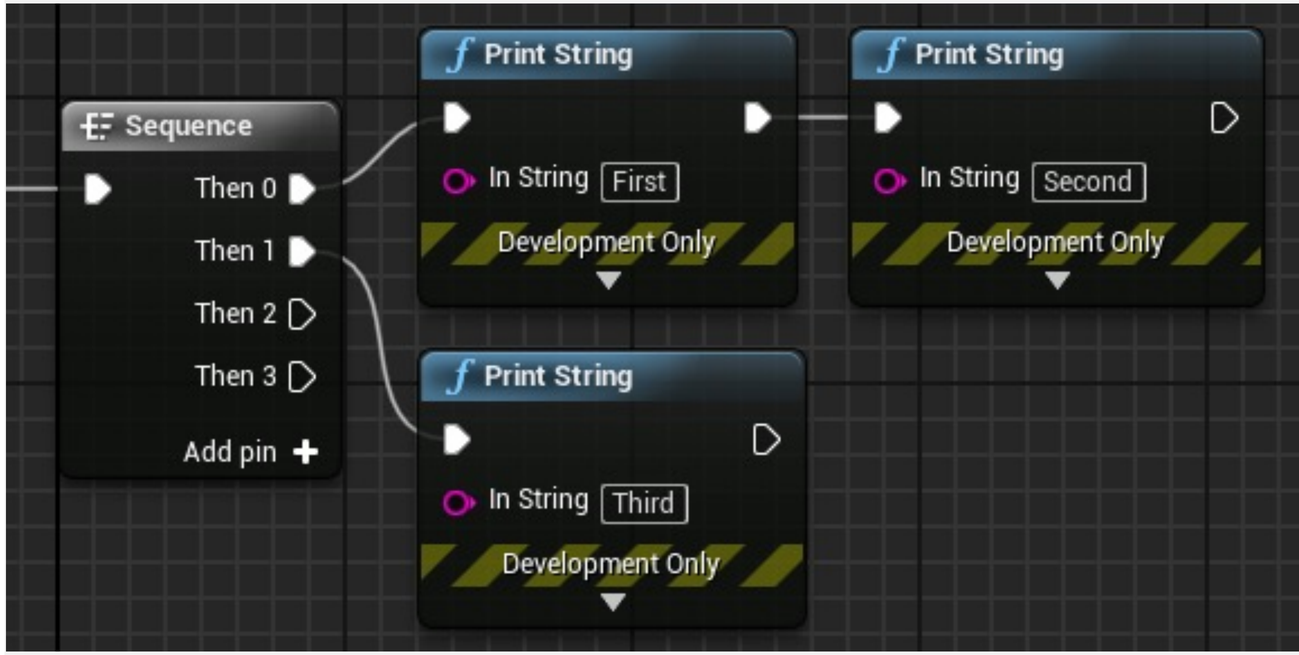
执行流程控制（Control Flow）

- 有一些节点负责控制蓝图的执行流
- 程序执行的三种基本控制流



序列（Sequence）节点

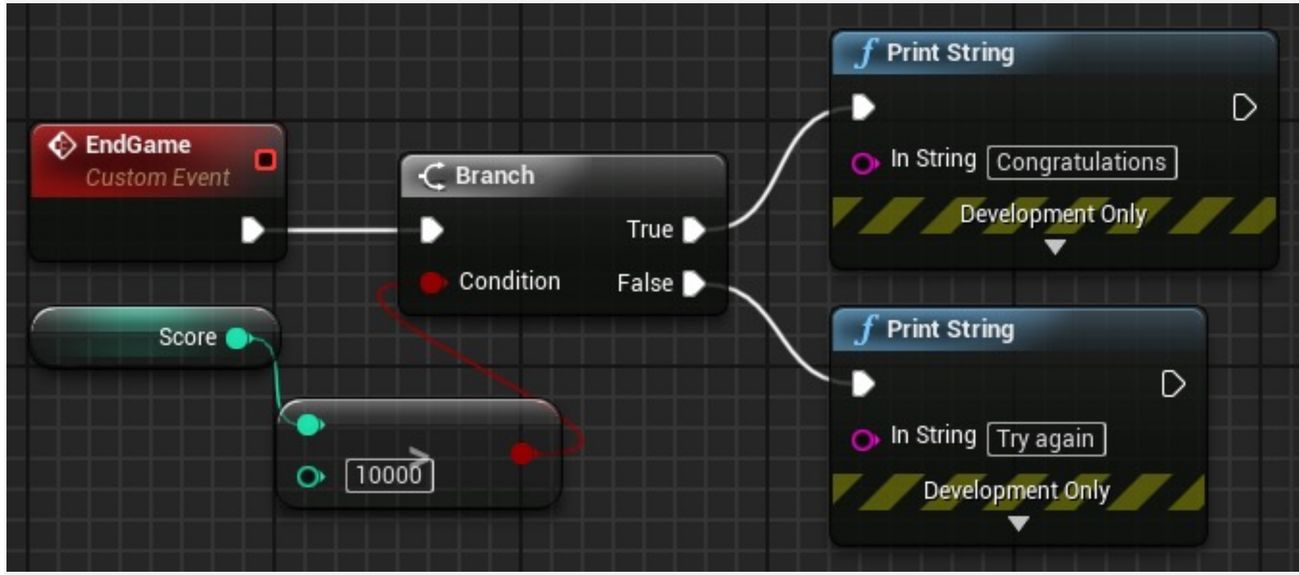
- 序列（Sequence）节点可以用于帮助组织其他蓝图操作。触发时，它按顺序执行与输出引脚相连的所有节点——即，先执行引脚Then 0的所有操作，然后执行引脚Then 1的操作，以此类推。
- 输出引脚可以使用“添加引脚+”（Add pin +）选项进行添加。要移除引脚，右键单击引脚，然后选择“移除执行引脚”（Remove execution pin）选项。



分支（Branch）节点

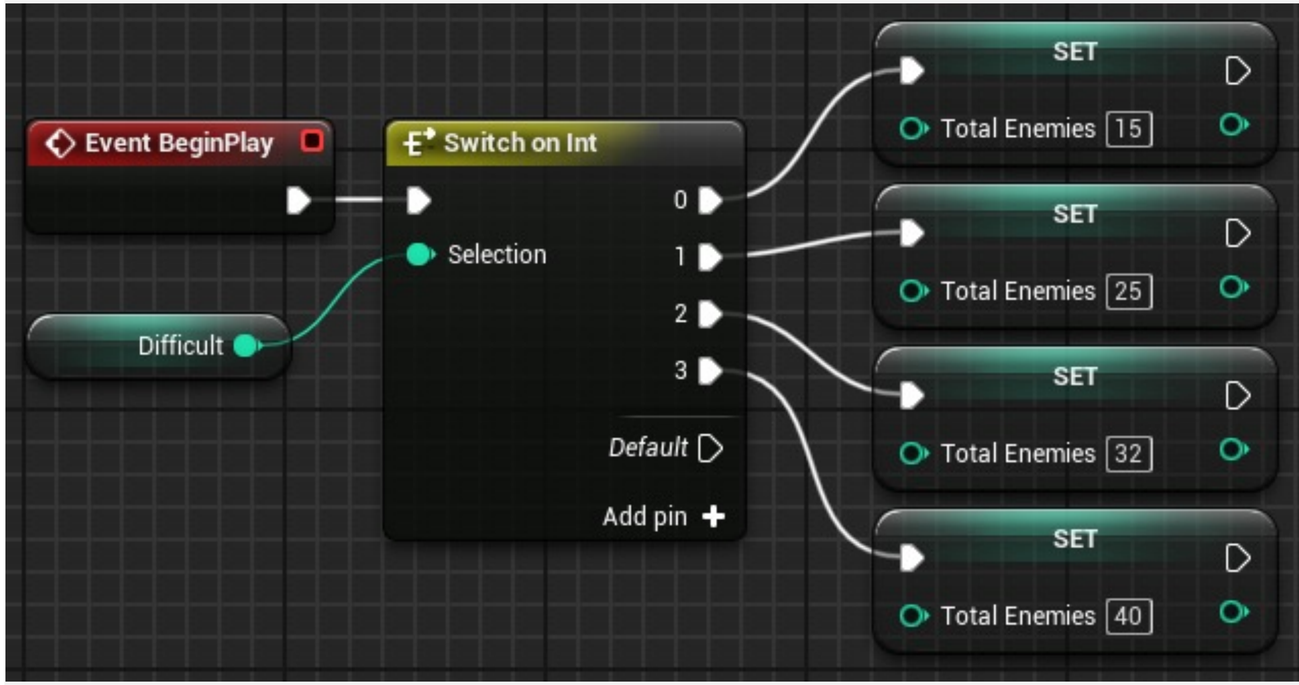
- 分支（Branch）节点根据布尔输入“条件”值（“true”或“false”）来引导蓝图执行的流向。

在右图中，游戏结束时会调用一个自定义事件。这个分支（Branch）节点用于测试分数是否大于“10000”。根据结果会显示不同的消息。



Switch On Int 节点

- Switch on Int 节点根据整数输入值确定执行流。
- 输出引脚可以使用“添加引脚+”（Add pin +）选项进行添加。
- 选择（Selection）：接收整数值，这个值确定将执行哪个输出引脚。如果未找到值，将执行“默认”（Default）引脚。

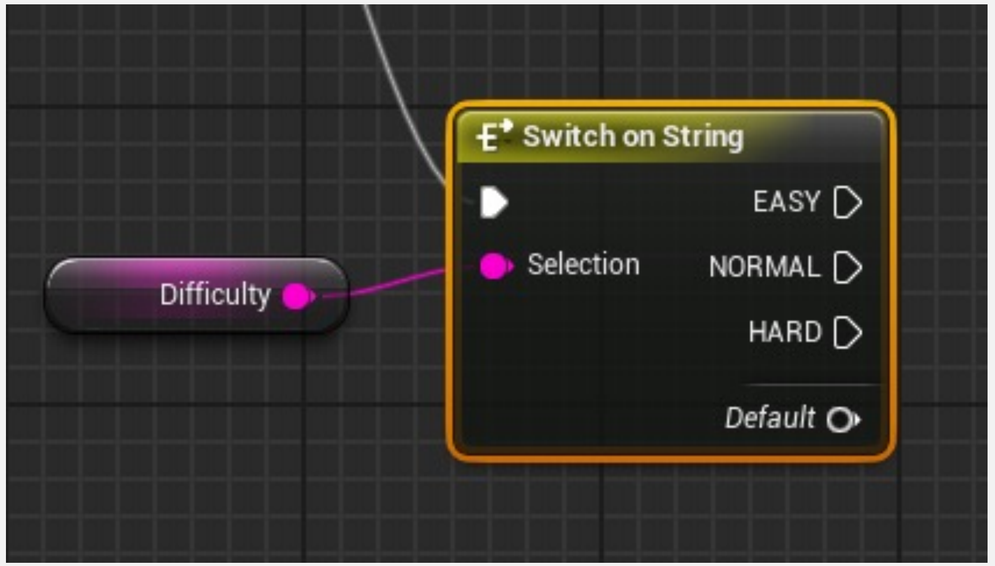


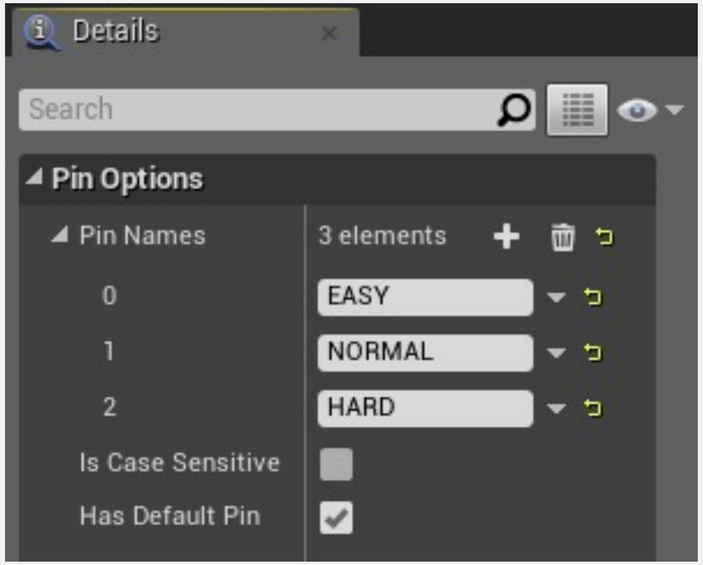
示例

- 在右侧示例中，游戏难度存储在整数变量“难度”（Difficult）中，其值范围是“0-3”。
- 敌人总数根据难度而设置。

Switch On String 节点

- Switch on String 节点根据字符串输入值确定执行流。
- 输入字符串与每个引脚名称对比，如果匹配，则执行这个引脚。
- 使用“细节”（Details）面板中的“引脚选项（Pin Options）>引脚名称（Pin Names）”下面进行配置。
- 选择（Selection）：接收用于确定输出的字符串值。

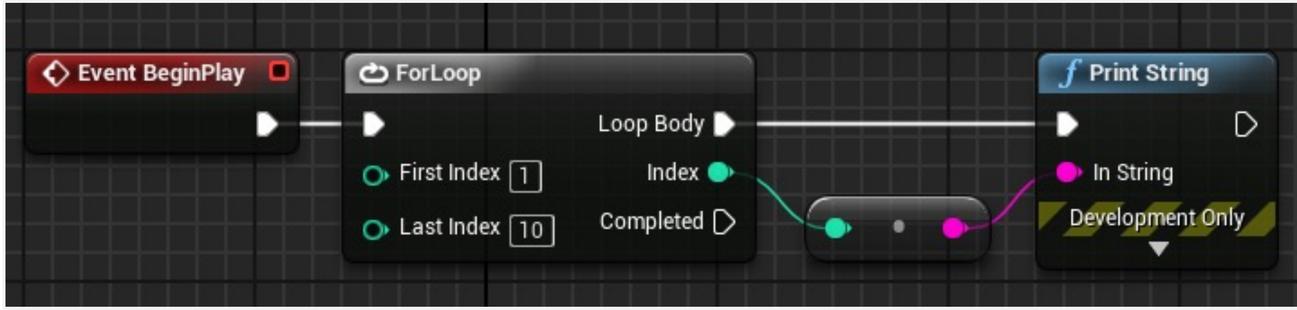


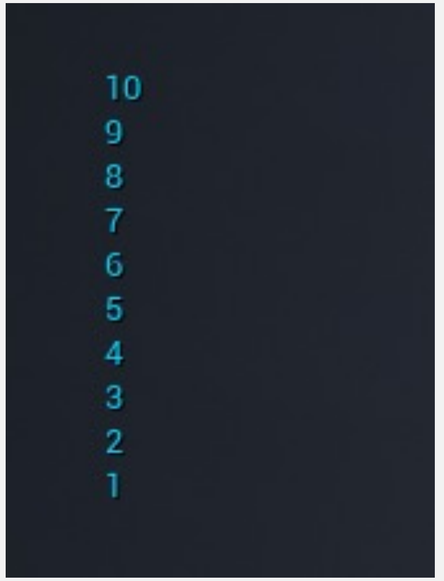


For Loop 节点

- ForLoop节点针对每个索引，执行与输出引脚Loop Body关联的一组操作。
- 当ForLoop节点完成执行时，将触发输出引脚“完成”（Complete）。

在右侧，ForLoop节点用于执行十次“打印字符串”（Print String）节点。ForLoop节点的索引输出引脚的值用作“打印字符串”（Print String）节点的输入。





小目标3

红色的门

小目标3

如果门是锁住的，则不会自动开启：

- 使用 **Construction Script** 把上锁的门自动变成红色

构造函数脚本

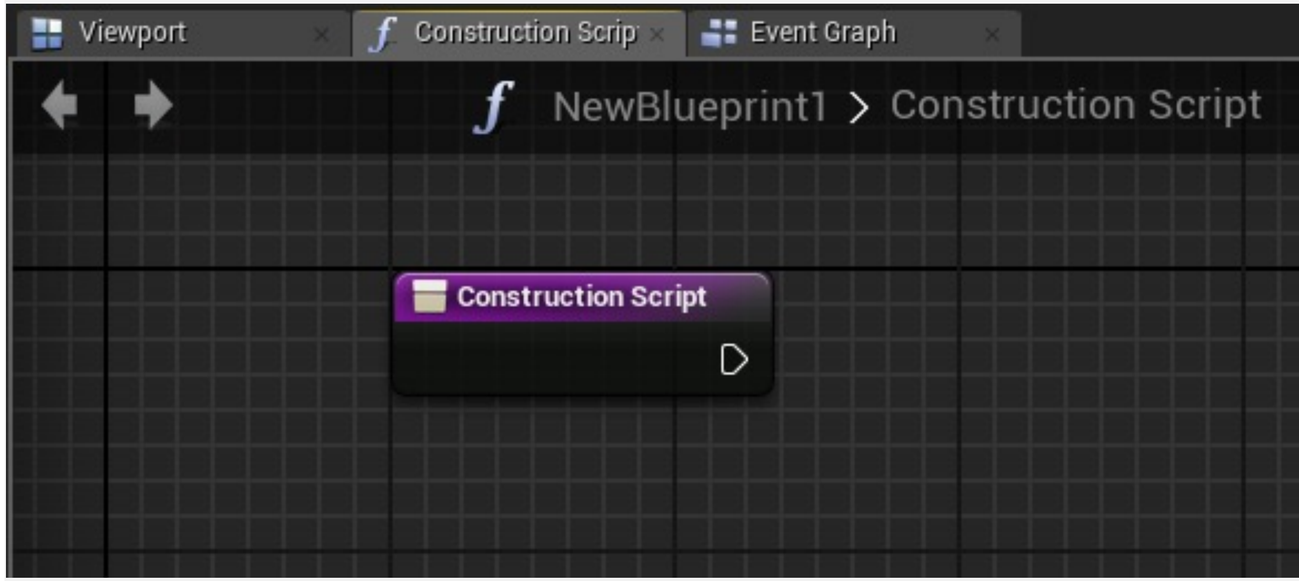
Construction Script

构造函数脚本

构造函数脚本是一个特殊的函数，当蓝图添加到关卡、蓝图属性发生更改或运行时产生实例对象时，会被引擎调用。

构造函数脚本有单独的图表，用来放置要执行的操作。

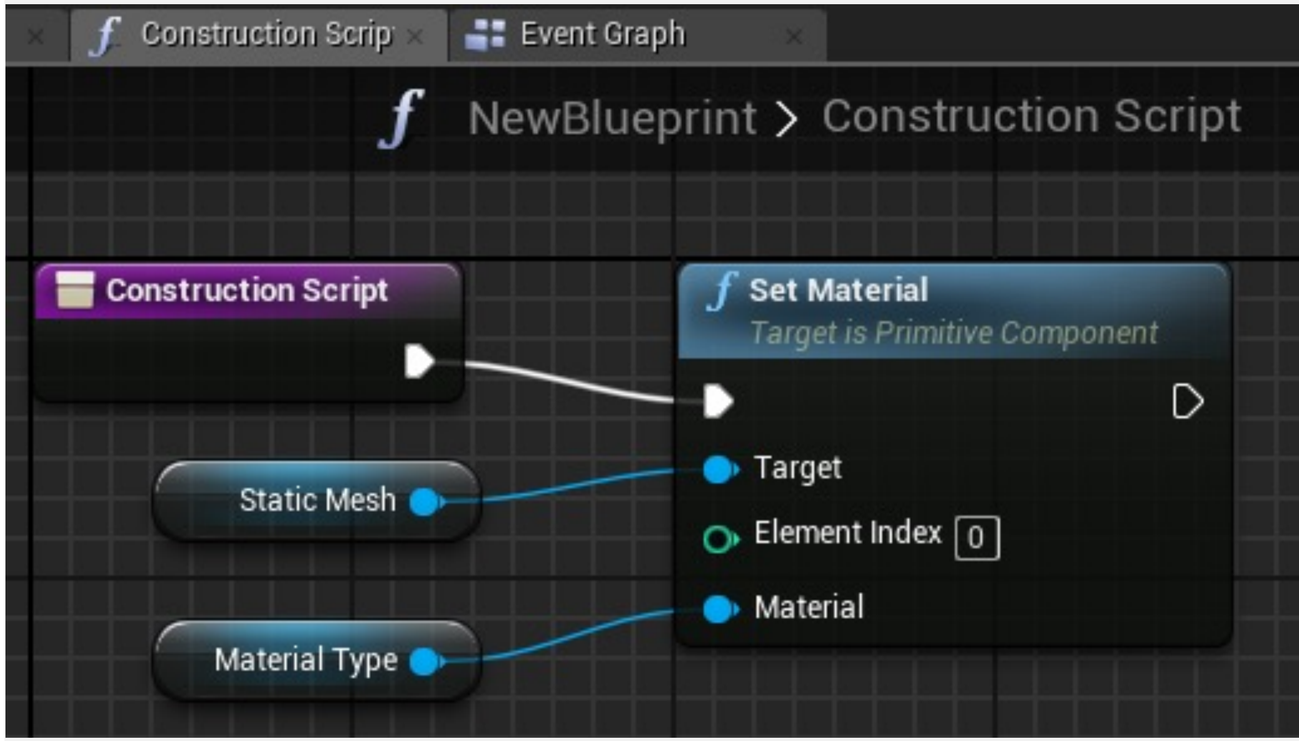
需要注意的是，构造函数脚本不会在游戏开始时对放置的Actor运行。



构造函数脚本：示例

右图所示的构造函数脚本（Construction Script）使用设置材质（Set Material）函数，根据可编辑变量材质类型（Material Type）中选中的材质来定义静态网格体（Static Mesh）组件的材质类型。

修改材质类型（Material Type）变量时，构造函数脚本（Construction Script）会再次运行，以使用新材质来更新对象。



小 目 标 4

场景中的道具

小目标4

- 使用简单的数学表达式产生浮动动画
- 玩家触碰道具后，自动拾取

运算符

Operators

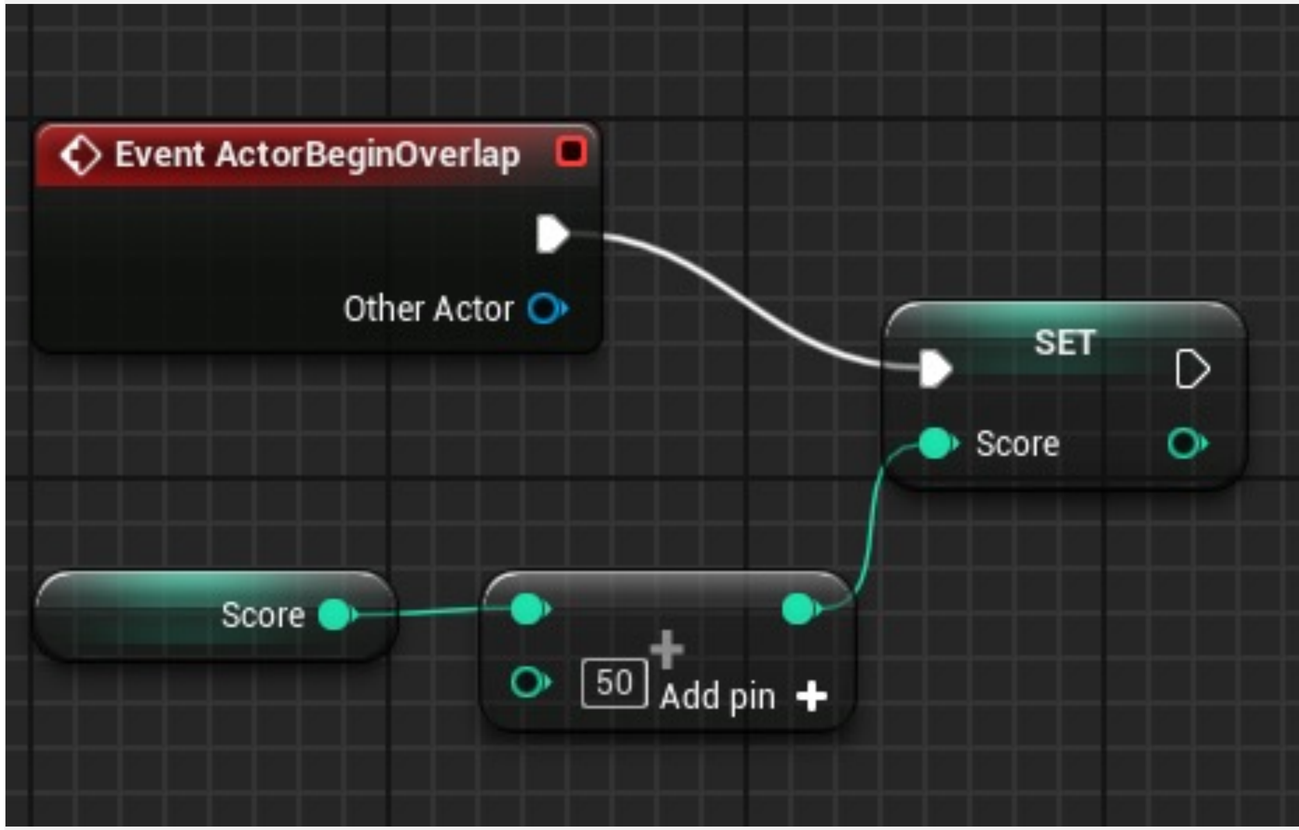
算术运算符(Arithmetic operators)

算术运算符（+、-、*、/）可以用于在蓝图中创建数学表达式。

右图显示一个简单的表达式，它向当前的分数（Score）变量添加值“50”，然后在分数（Score）变量中设置新计算的值。

“+”运算符在左侧获得两个输入值，然后在右侧给出运算结果。要使用两个以上输入值，只需单击“添加引脚”（Add pin）选项。

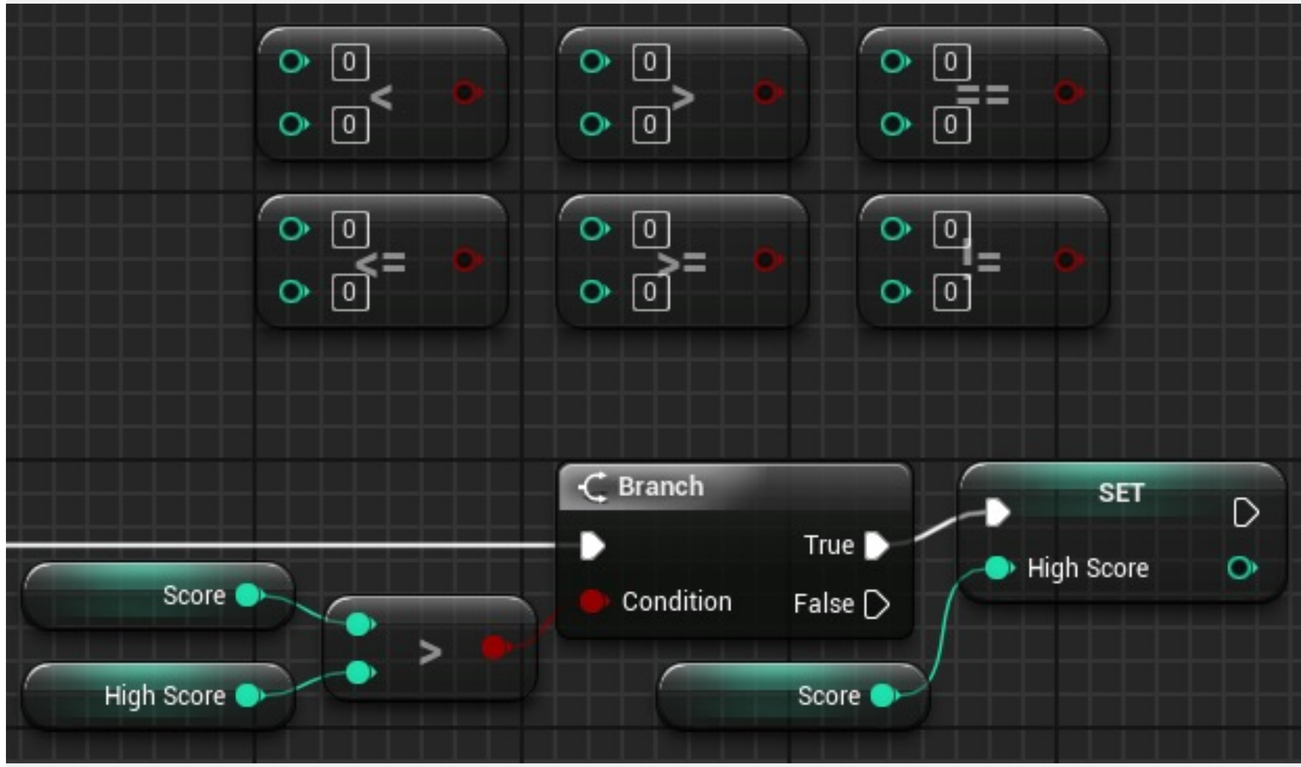
输入值可以直接输入到节点中，或者可以从变量获取。



关系运算符(Relational operators)

关系运算符比较两个值，然后返回布尔值（“true”或“false”）作为比较结果。

右图显示了使用关系运算符和使用分支（Branch）节点的示例。游戏结束时，当前玩家的分数（分数（Score）变量）将与最高纪录游戏分数（高分（High Score）变量）进行比较。如果玩家分数更高，则在高分（High Score）变量中存储分数（Score）变量的值。

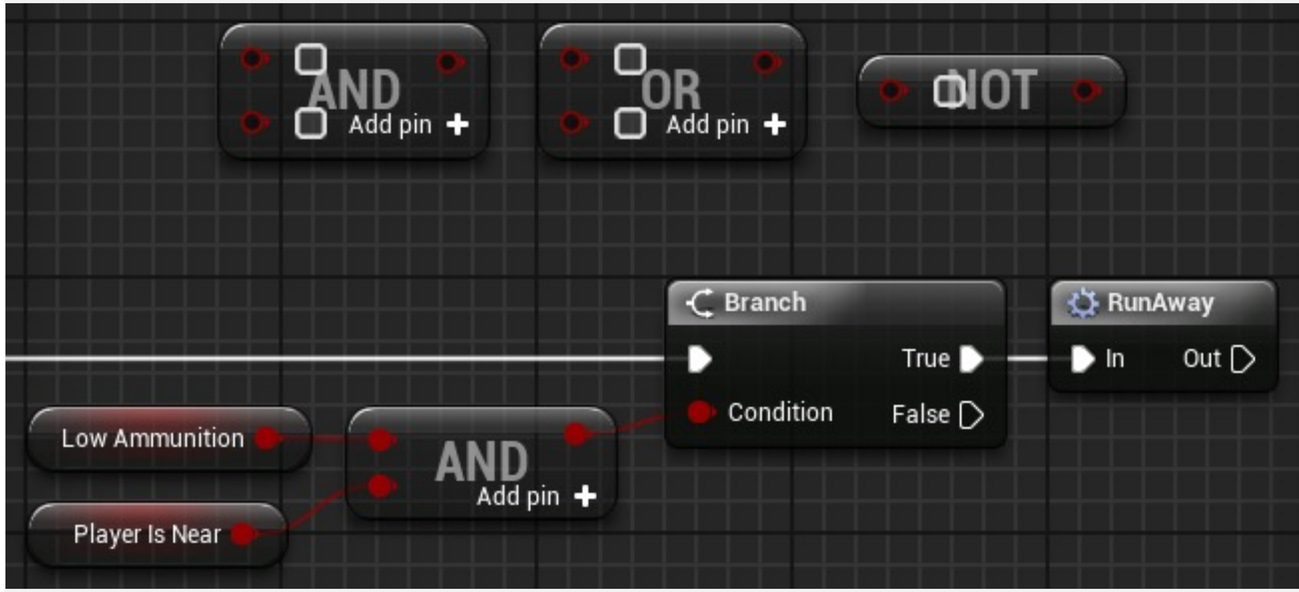


逻辑运算符(Logical Operators)

逻辑运算符在两个布尔值之间执行运算，然后返回布尔值（“true”或“false”）作为运算结果。主要逻辑运算符包括：

- AND：如果所有输入值为“true”，则返回值“true”。
- OR：如果任意输入值为“true”，则返回值“true”。
- NOT：仅获取一个输入值，结果将为相反值。

右侧示例模拟了游戏中简单的敌人决定。如果敌人弹药不足（弹药低（Low Ammunition）变量），而玩家在附近（玩家在附近（Player Is Near）变量），那么敌人决定逃跑。



数学表达式

Math Expression

数学表达式节点

数学表达式（Math Expression）节点是一类特殊类型的节点，使用指定的数学表达式生成子图。

要使用它，在快捷菜单中选择“添加数学表达式...”（Add Math Expression...）。右图显示了使用表达式“x + y”的简单示例。

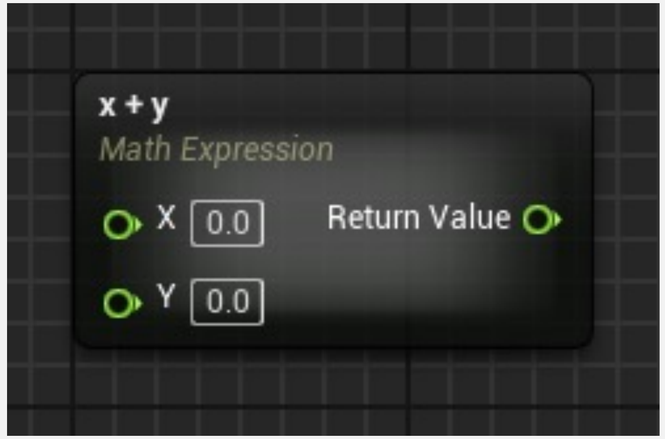
输入

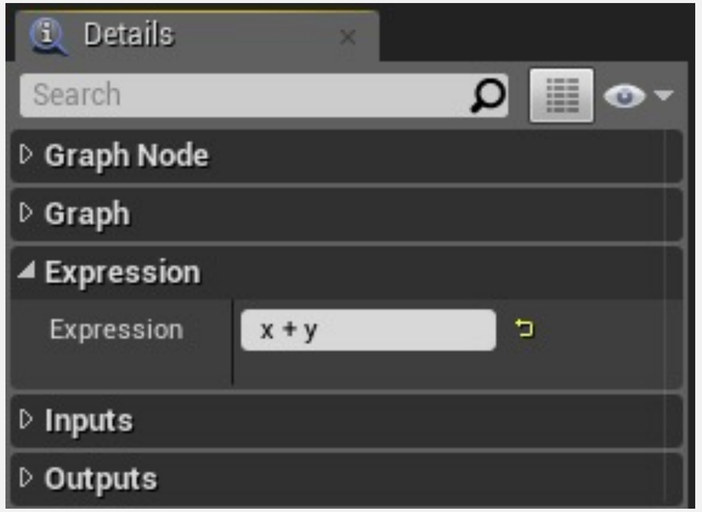
•表达式（Expression）：将要分析的表达式。

•“表达式”中定义的参数（Parameters defined in “Expression”）：对于表达式中的每个变量名称，将生成新输入参数。

输出

•返回值（Return Value）：输出表达式的结果。



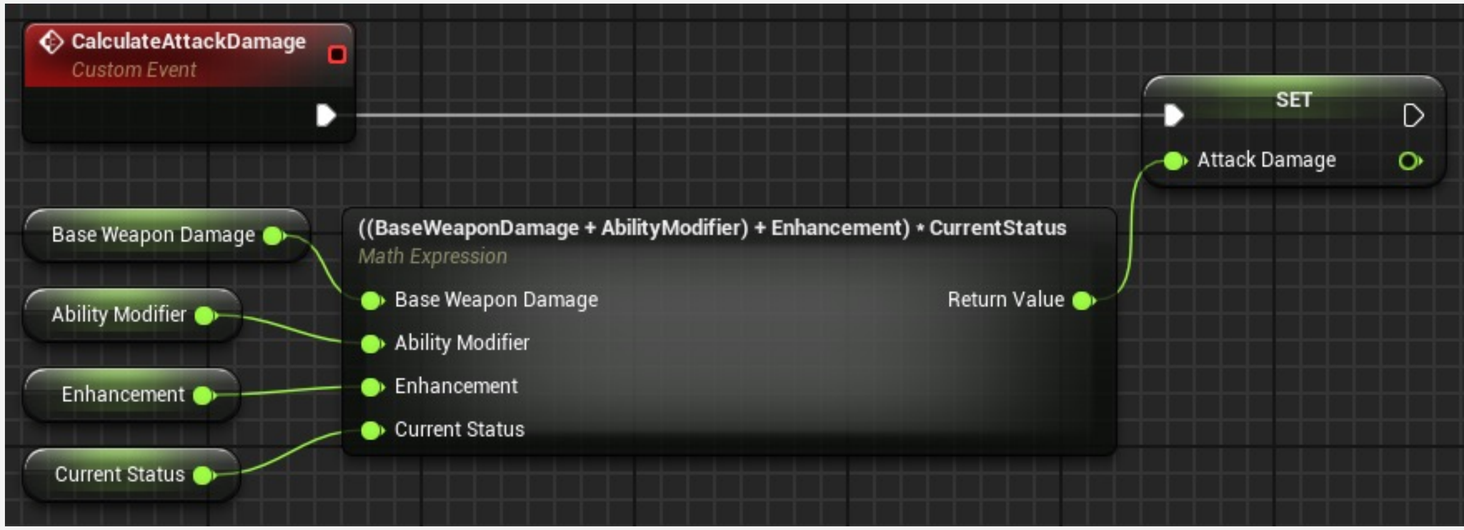


数学表达式节点：示例

在右侧示例中，计算攻击伤害（CalculateAttackDamage）事件将计算攻击伤害，并将结果存储在变量中。它使用Math Expression节点和以下表达式：

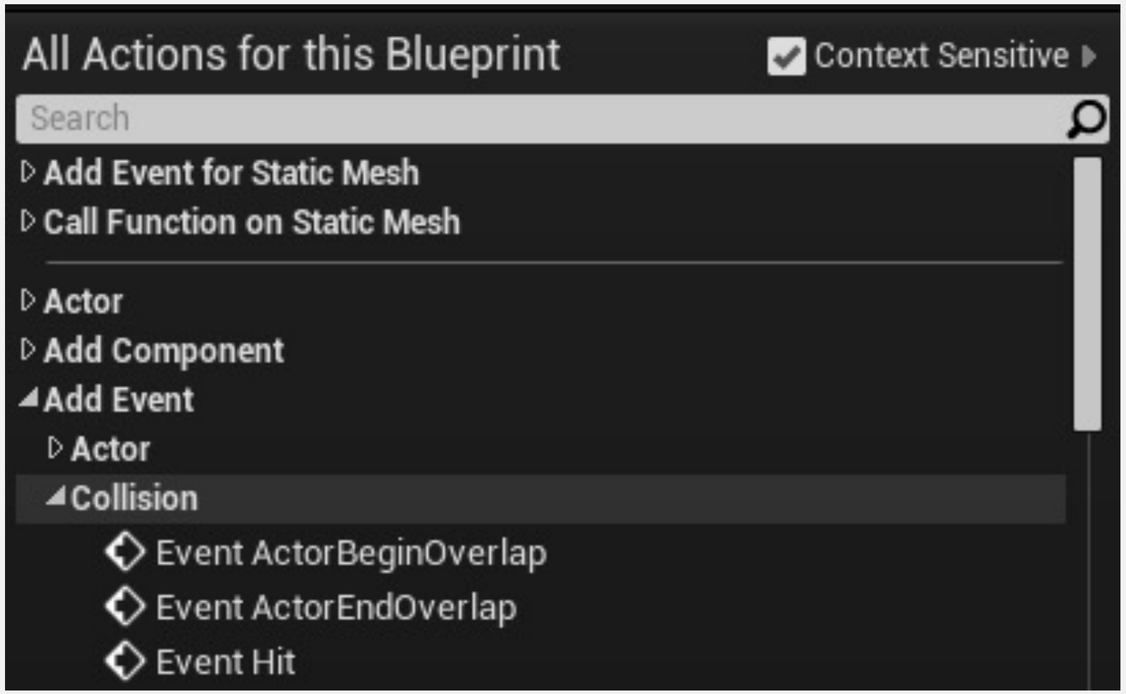
((BaseWeaponDamage + AbilityModifier) + Enhancement) * CurrentStatus

输入参数根据该表达式生成。



Actor 事件响应

Actor Events



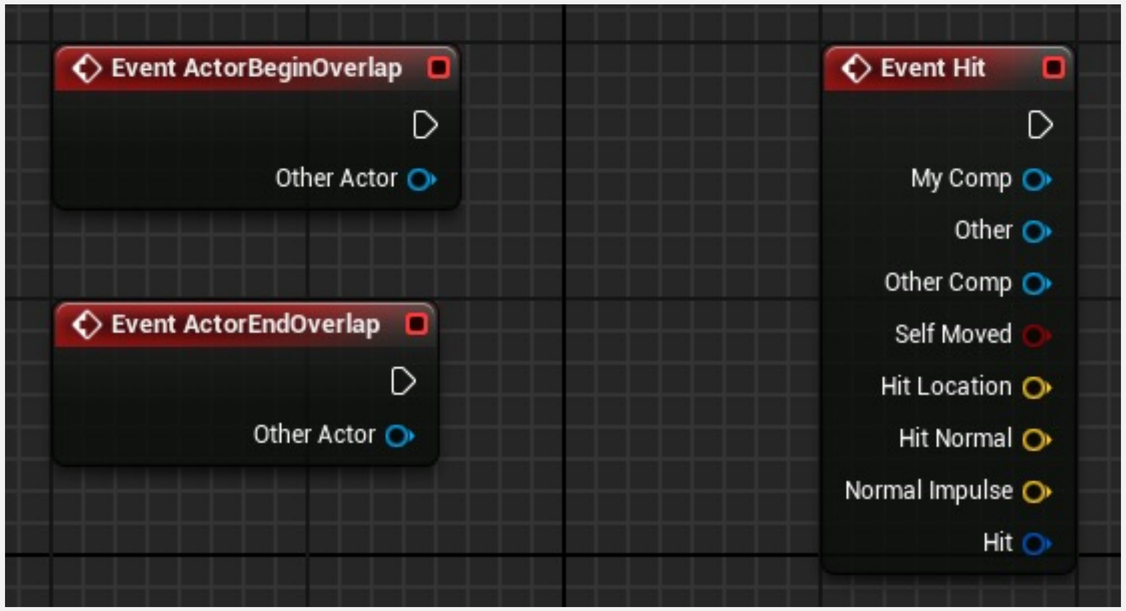
碰撞事件(Collision events)

当两个Actor碰撞或重叠时会发生碰撞事件。

当两个Actor开始重叠，并且两个Actor的“生成重叠事件”（Generate Overlap Event）属性都设置为“true”时，则发生“Actor开始重叠”（ActorBeginOverlap）事件。

当两个Actor停止重叠时，将执行“Actor结束重叠”（ActorEndOverlap）事件。

如果处于碰撞中的任一Actor的“模拟生成撞击事件”（Simulation Generates Hit Events）属性设置为“true”，则执行“撞击”（Hit）事件。



小 目 标 5

道具收集

小目标5

场景中有多种道具，玩家可以收集：

- 设计道具相关的枚举和数据结构
- 在 **Game Mode** 中记录玩家收集的道具类型

枚举类型

Enumeration

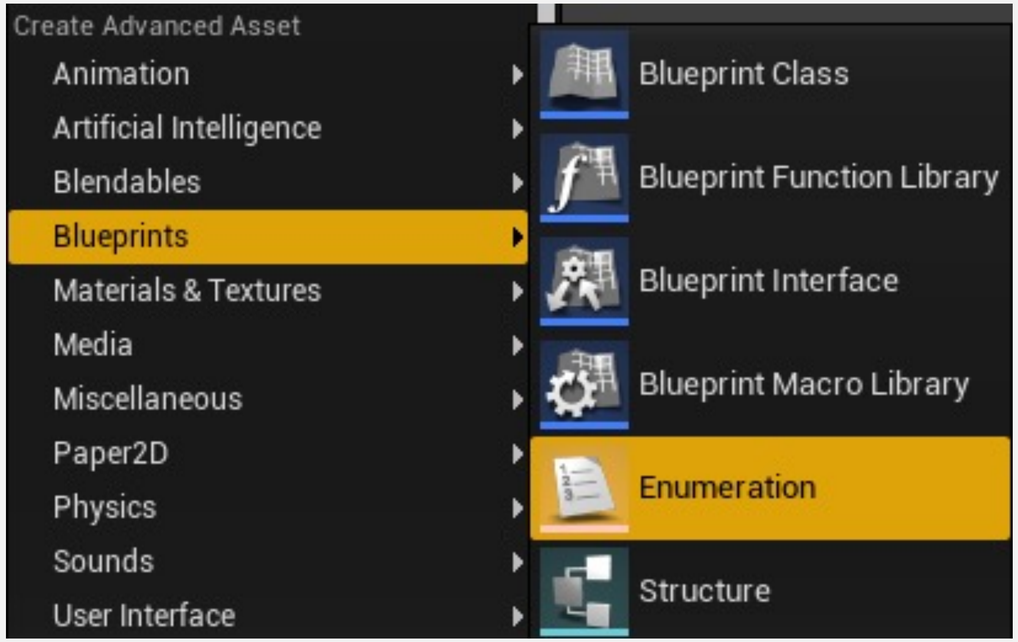
枚举（enum）

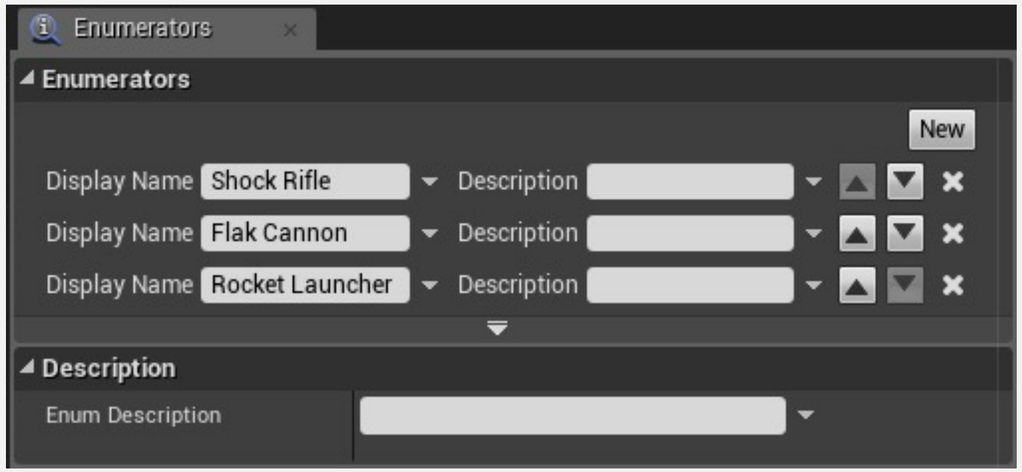
枚举（enum）是一组常量，它们拥有有意义的名称，用于指定一个变量可以拥有的所有可能值。

要创建新的枚举类型，单击内容浏览器中的绿色“新增”（Add New）按钮，然后在“蓝图”（Blueprints）子菜单中选择“枚举”（Enumeration）。

双击之前创建的枚举进行编辑。

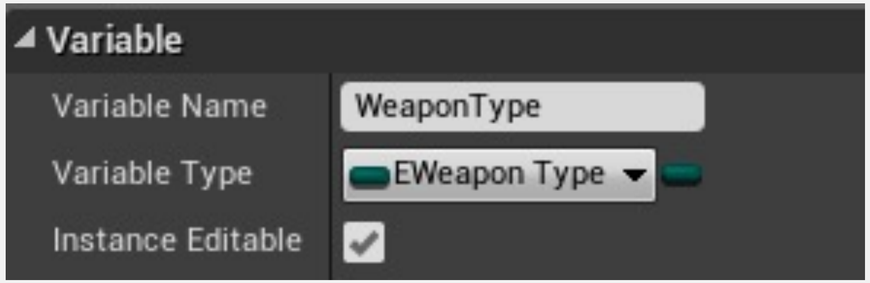
单击“新增”（New）按钮来添加属于列举一部分的名称。





枚举：创建变量

- 定义好枚举类型之后，就可以在变量类型列表中找到它
- 编辑器会自动将枚举类型作成下拉列表，供选择

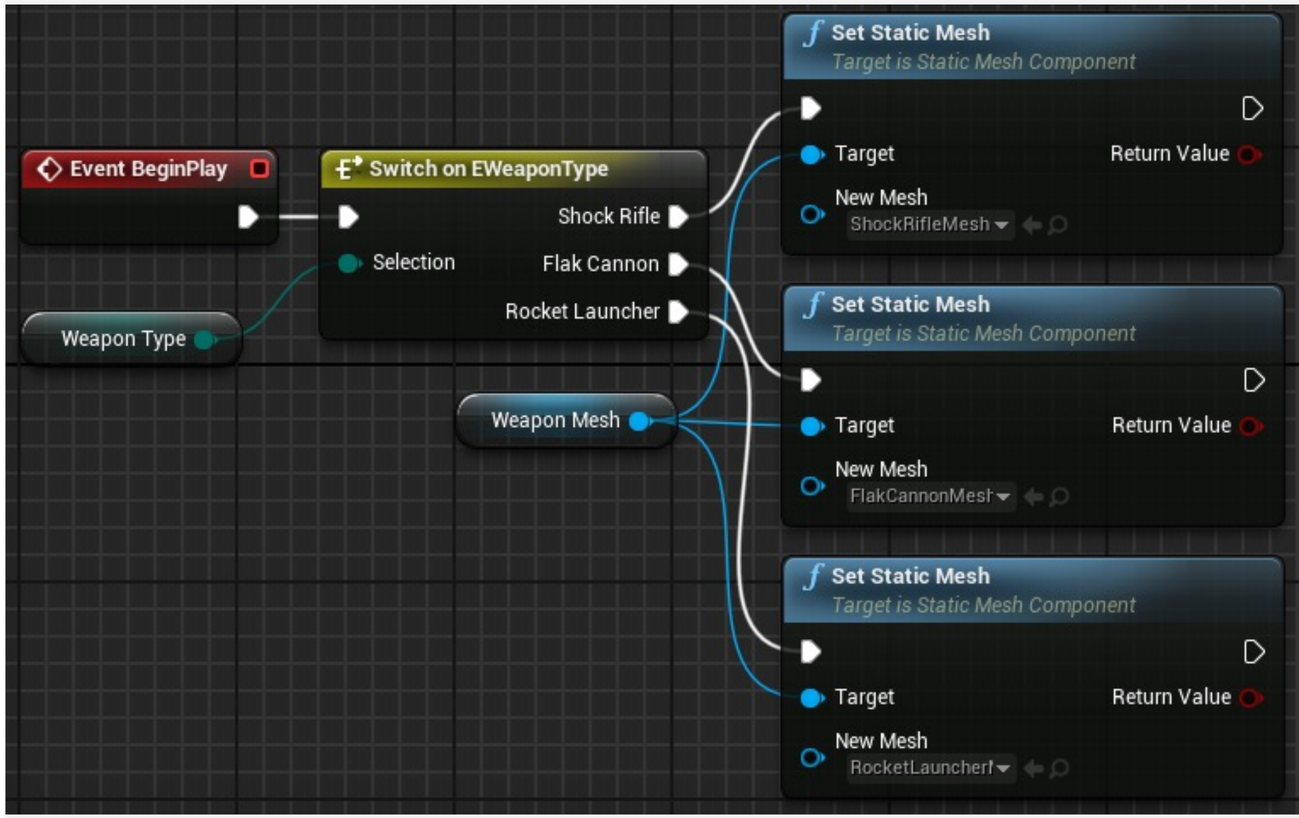


枚举与 Switch On

有一类开关节点根据列举值来确定执行流。

在右图中，“武器类型”（Weapon Type）是一个列举变量，“武器网格体”（Weapon Mesh）是一个静态网格体组件。

该静态网格体将根据武器类型进行设置。



游戏模式

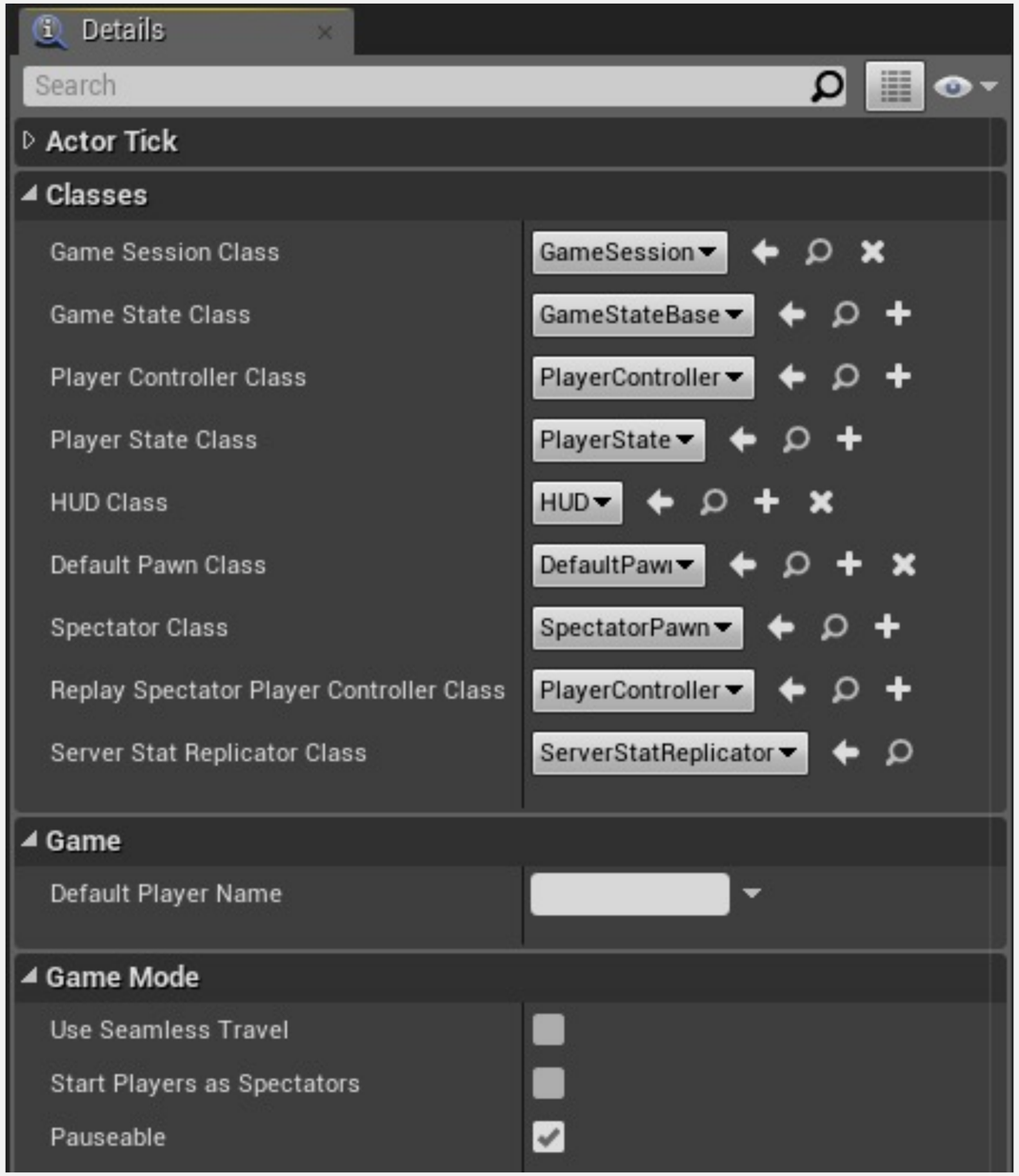
Game Mode

游戏模式(Game Mode)

- Game Mode类用于定义游戏的规则。
- 游戏模式还指定将用于创建Pawn、Player Controller、Game State、HUD和其他类的默认类，如右图所示。
- 想象一下《虚幻竞技场》：夺旗模式、死亡竞赛模式等

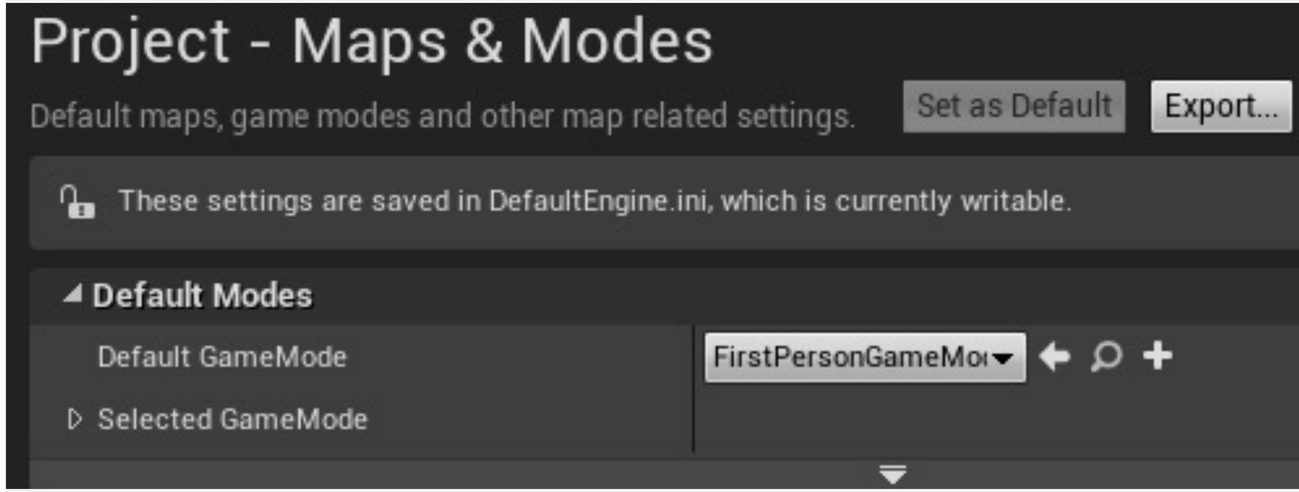
每个关卡都可以有不同的游戏模式。如果不为关卡指定游戏模式，则将使用已经为项目设置好的游戏模式。

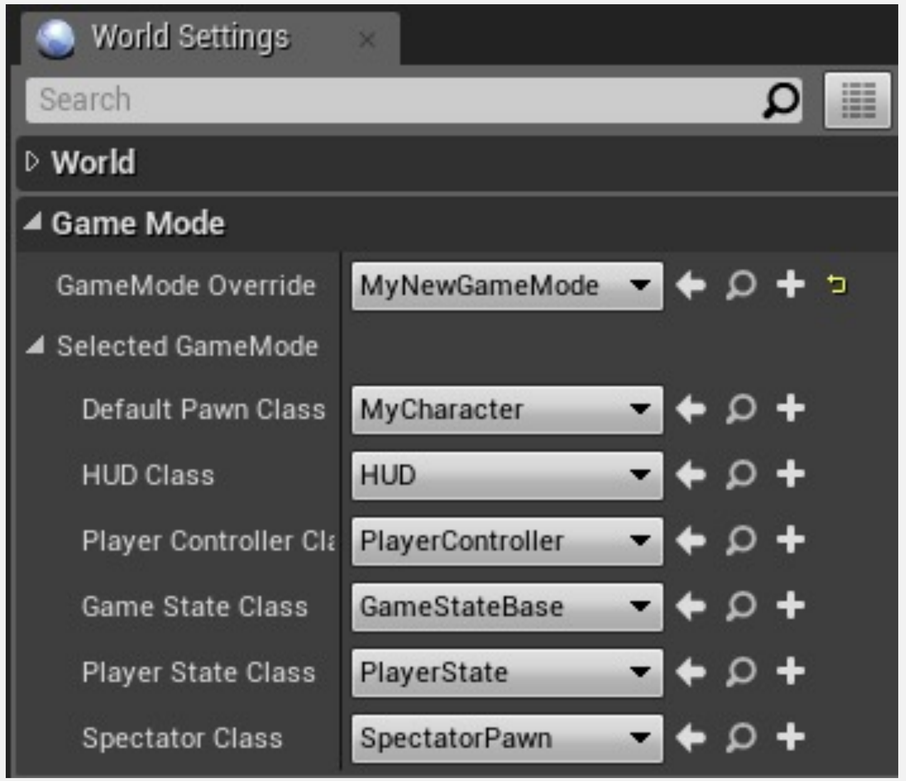
在多人游戏中，游戏模式仅存在于服务器上，不会复制到客户端。



指定游戏模式

- 为项目指定默认游戏模式
- 指定关卡的游戏模式，关卡的游戏模式将覆盖项目的默认游戏模式。





数据结构

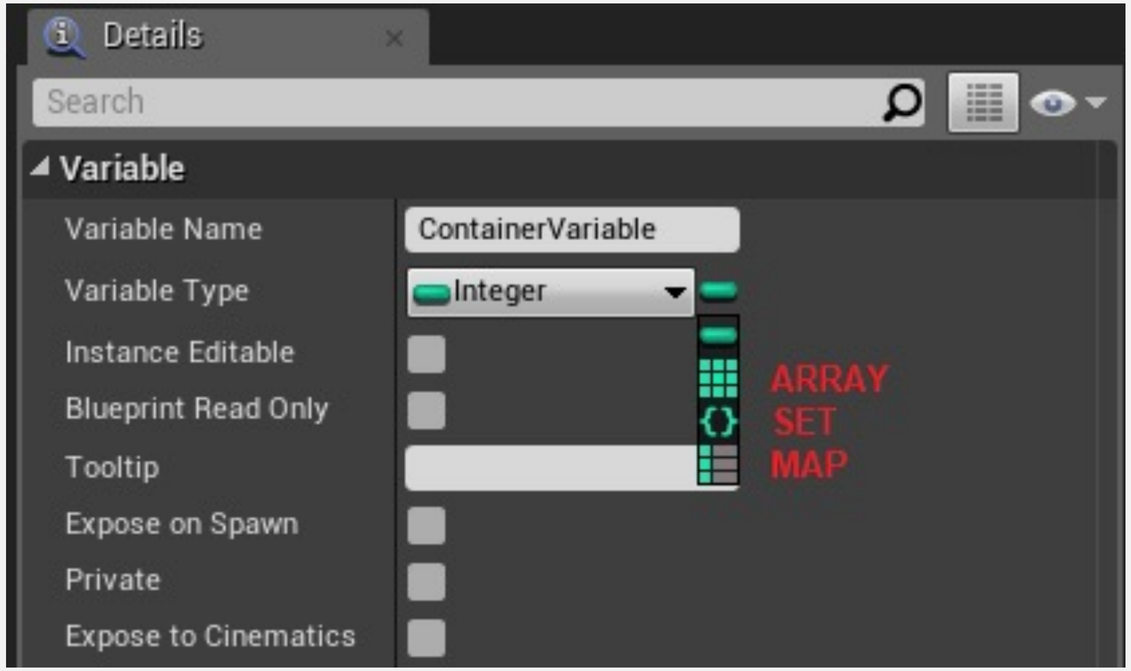
Data Structures

数组、Set 和 Map

“变量类型”（Variable Type）属性包含一个按钮，用于将变量转换为容器。

容器可以存储同一类型的多个元素。以下所列是可用容器的类型。

- 数组（Array）：可以使用索引值访问的值的有序列表。
- Set：值的无序集合，不允许重复值。
- Map：使用键-值对定义每个条目的列表，不允许重复键值。



数组(Array)

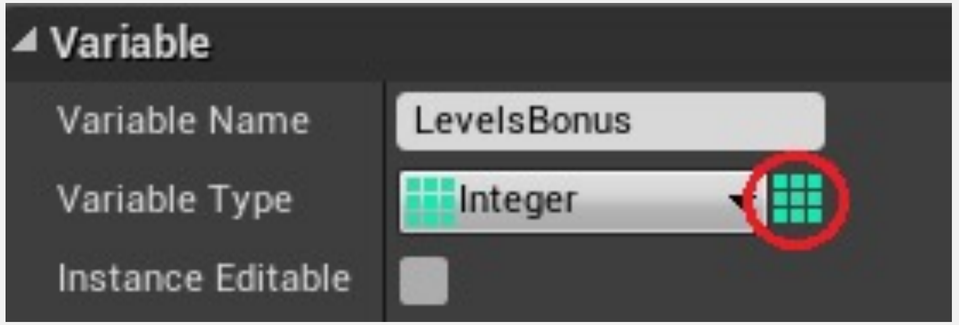
数组是一个有序列表，您通过使用基于整数的索引来设置和获取数组元素。

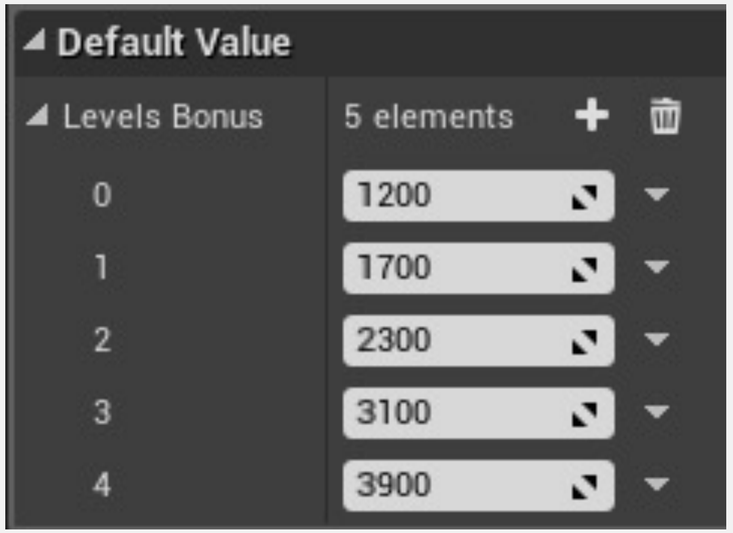
通过使用数组，可以将同一类型的变量分组在一起。在蓝图中创建数组十分简单。

首先，创建新变量，并选择所需类型。

单击“变量类型”（Variable Type）下拉菜单旁边的图标，并选择“数组”（Array）（见右上图）。

编译蓝图后，可以在数组元素中填充默认值，如右下图所示。

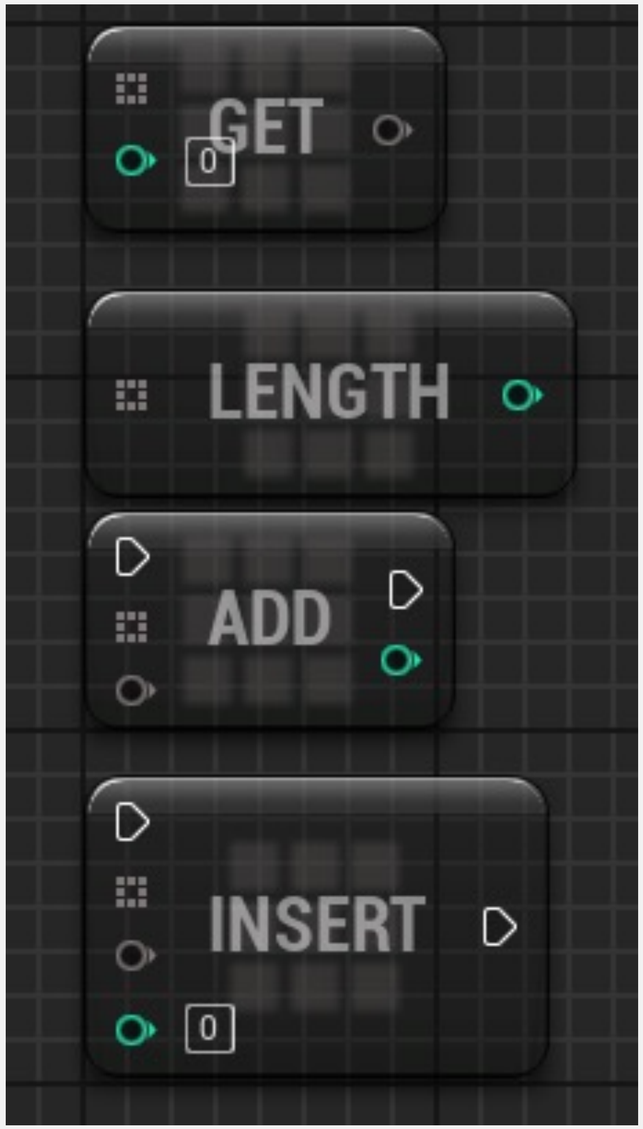




数组：主要节点

与数组有关的主要节点如下：

- 获取（Get）：返回所用索引指定的位置处的元素。
- 长度（Length）：返回数组元素的数量。
- 添加（Add）：在数组结尾添加新元素。
- 插入（Insert）：在索引参数指定位置处添加新元素。



数组遍历：For Each Loop

ForEachLoop节点接收数组作为输入参数，并针对可以从“数组元素”（Array Element）输出引脚获取的每个数组元素，执行一组与Loop Body输出引脚关联的操作。之后，执行流被引导至“完成”（Completed）输出引脚。

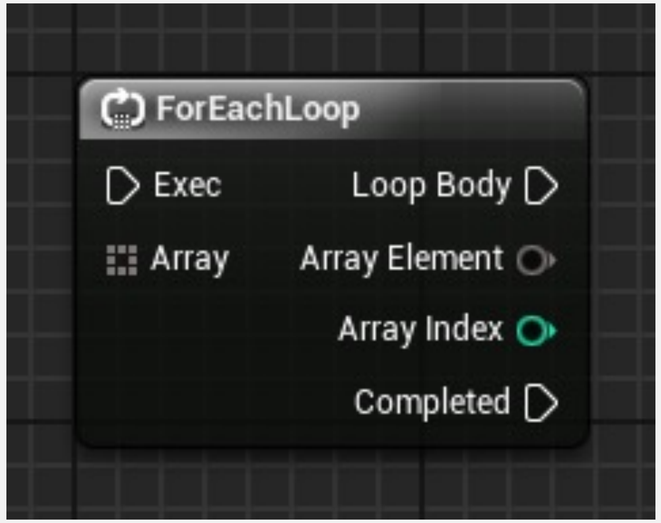
输入

•数组（Array）：接收一个数组，其中包含将在循环中使用的元素。

输出

•数组元素（Array Element）：输出对当前数组元素的引用。

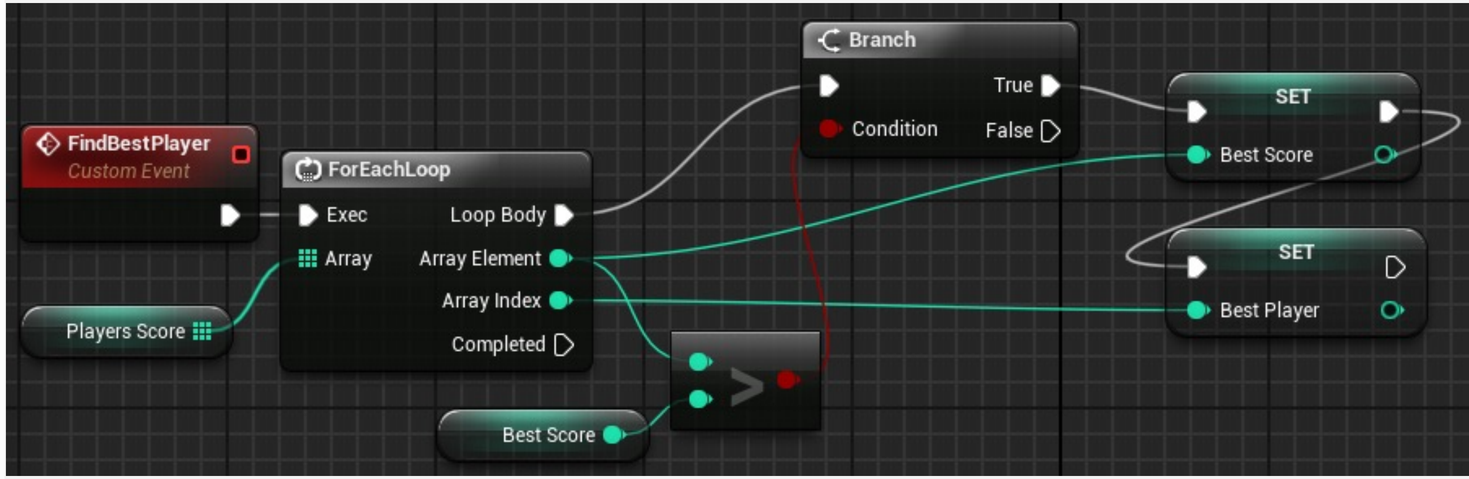
•数组索引（Array Index）：输出当前数组元素的索引。



数组的遍历：示例

在右侧示例中，使用了ForEachLoop节点迭代包含玩家分数的数组。

对于每个值，进行一次测试，以确认它是否表示最高分。如果为“true”，则值存储在“最高分”（Best Score）变量中，玩家索引存储在“最厉害玩家”（Best Player）变量中。



Set

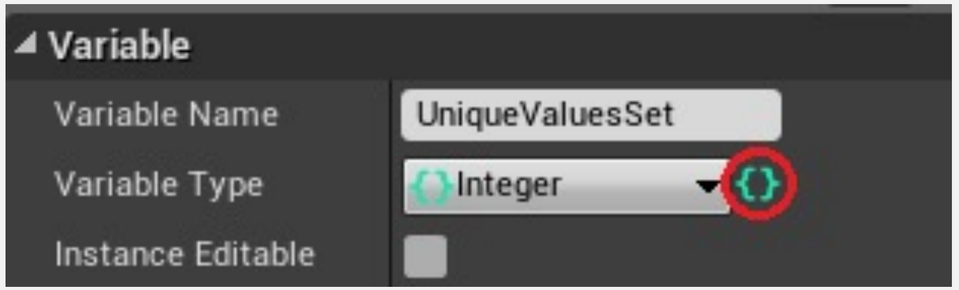
一个 Set 包含一类容器，类似于数组，但与数组不同的是，集合是无序元素列表，根据值搜索其中的元素。没有索引。它可以用来将同类型的变量分组到一起。一个重要的差异是集合不允许存在重复元素。

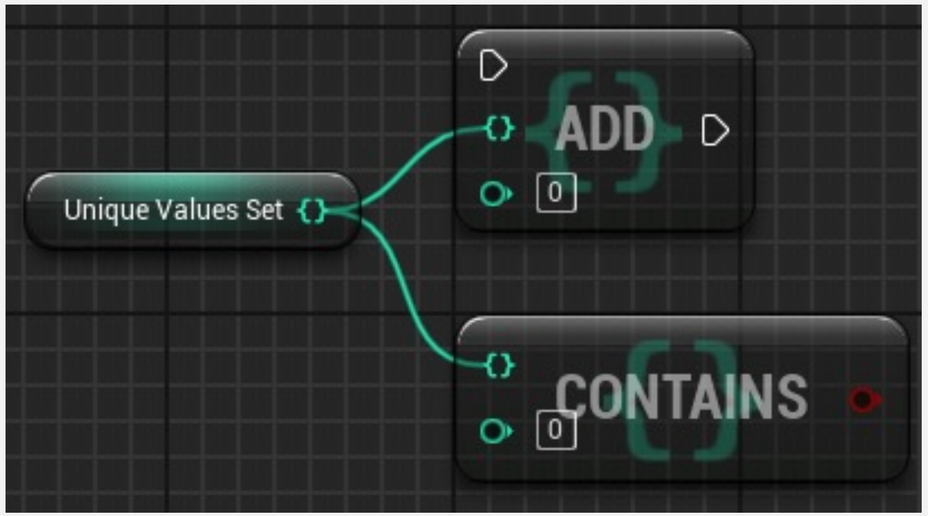
用于查找项目的关键值就是项目本身。

要将变量定义为集合，单击“变量类型”（Variable Type）下拉菜单旁边的图标，并选择“集合”（Set）（见右上图）。

下面是与集合有关的一些常见节点：

- 添加（Add）：向集合添加项目。
- 包含（Contains）：检查以确认集合是否包含指定项目。
- 移除（Remove）：从集合移除项目。
- 长度（Length）：返回集合中的项目数量。





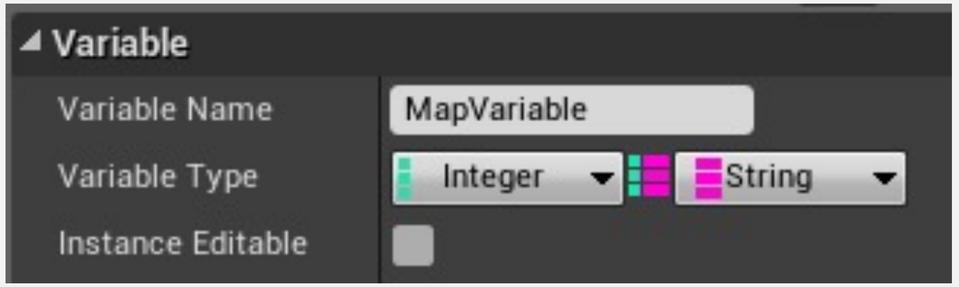
Map

还有一个类型的容器叫做 Map 。要将变量定义为 Map，单击“变量类型”（Variable Type）下拉菜单旁边的图标，并选择 “Map” 。

Map 的每个元素的值都有关联的键。Map 是无序的，使用键值进行搜索。在右上图所示的 Map 中，键类型是“整数”，值类型是“字符串”。

Map 的键值必须唯一。

右下图是一个 Map 示例，其中数字与游戏项目的名称关联。



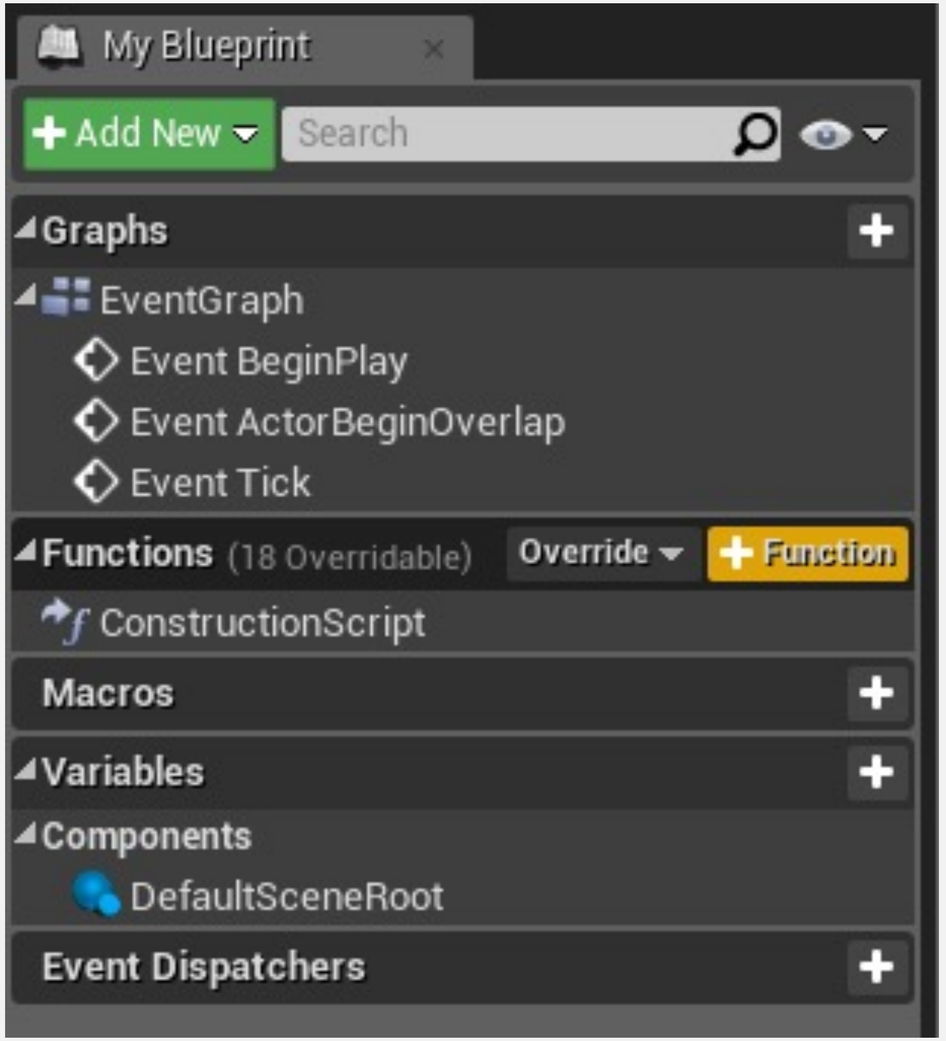
函数

Functions

函数(Functions)

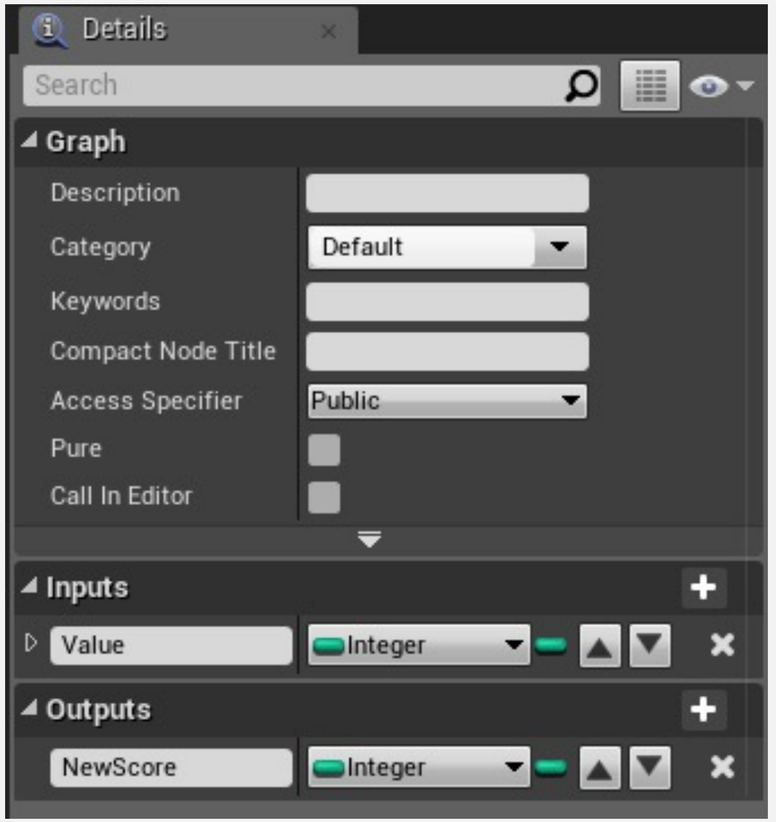
- 函数是脚本逻辑组织的最基本、最好用的单元
- 使用函数可以把一系列操作汇总在一个单元里，方便外部直接使用
- 函数可以有输入和输出参数

要创建函数，前往蓝图编辑器中的“我的蓝图”（My Blueprint）面板，单击“函数”（Functions）类别中的“+”符号。



函数：输入和输出

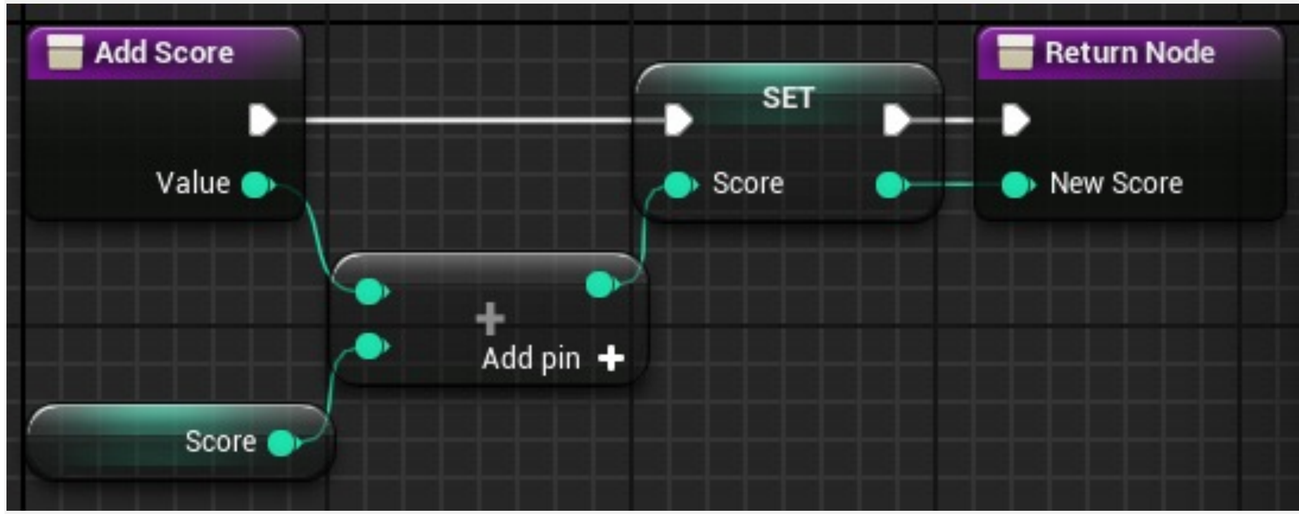
- 输入参数是可以传递给函数的值。
- 输出参数是可以从函数返回的值。
- 要添加输入或输出参数，在“我的蓝图”（My Blueprint）面板中选择函数，并使用“细节”（Details）面板。



示例

右图所示的函数有一个输入参数“值”（Value），它添加到分数（Score）变量。

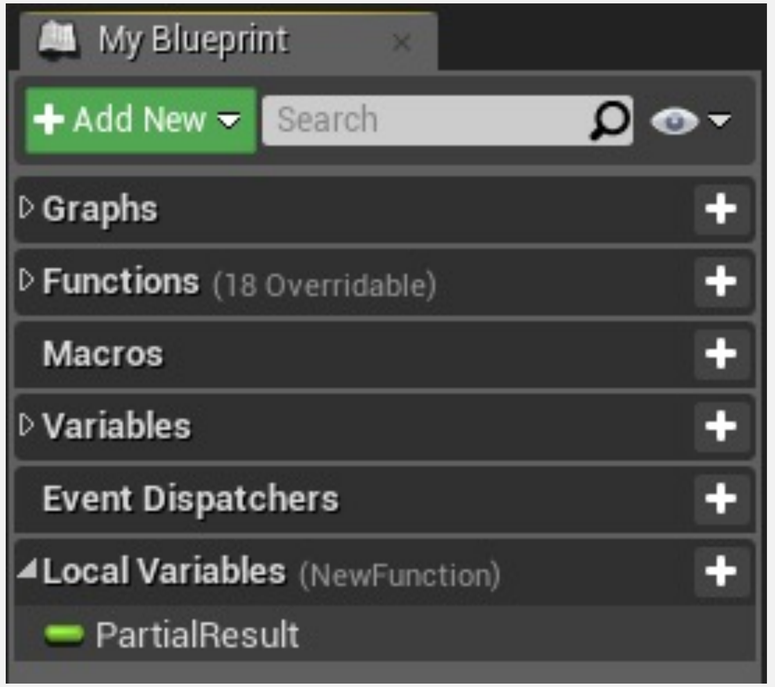
求和结果设置在分数（Score）变量中，然后通过输出参数“新分数”（New Score）返回。



函数：局部变量

- 函数可以拥有仅在函数内部可见的局部变量。
- 局部变量非常有助于实现复杂函数，不会与蓝图的其他变量混淆。

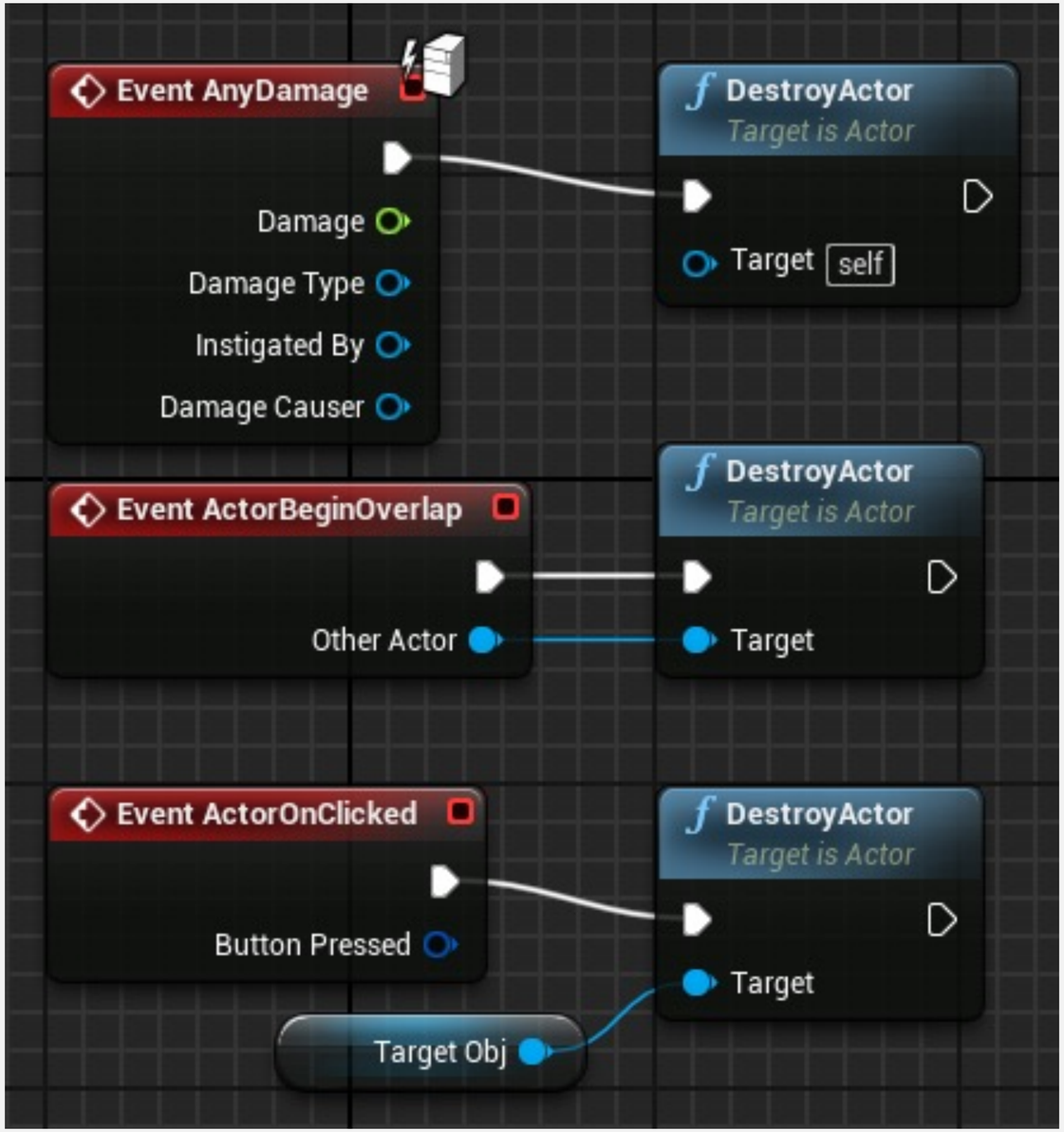
要创建局部变量，双击函数进行编辑，然后查看“我的蓝图”（My Blueprint）面板。在面板底部，您将找到一个名为“局部变量”（Local Variables）的类别，且函数名称用括号括起。单击“局部变量”（Local Variables）类别中的“+”按钮。



函数：目标（Target）参数

- Target（目标）参数是一个常见参数，表示将通过函数调用修改的对象。
- 该参数的默认值是“self”，这是指当前正在执行的这个对象的引用。

右图显示了 DestroyActor 函数的 Target 参数的不同用法。



小 目 标 6

随机生成道具

小目标6

- 自动的“随机”生成一些道具
- 使用关卡蓝图

关卡蓝图

Level Blueprint

回顾：主要蓝图类型

关卡蓝图(Level Blueprint)

关卡蓝图是一种特殊类型的蓝图，属于关卡。它用于定义关卡中的特定事件和操作。

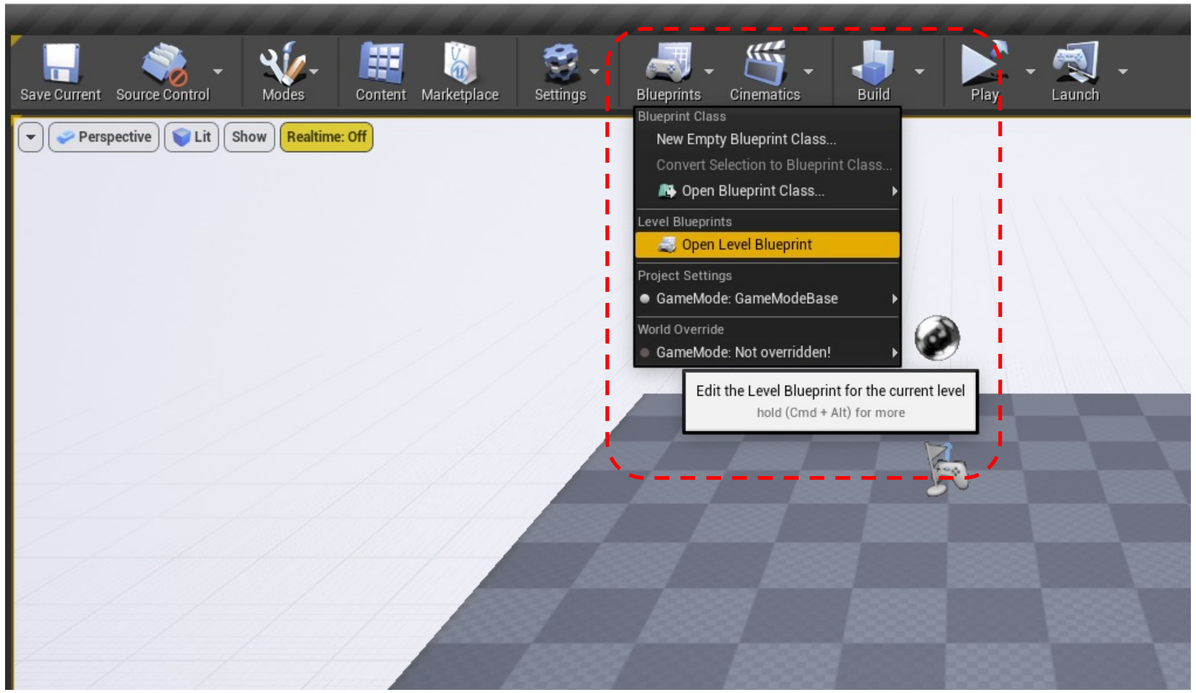
关卡蓝图可以用于与蓝图Actor类互动，以及管理某些系统，如过场动画和关卡流送。

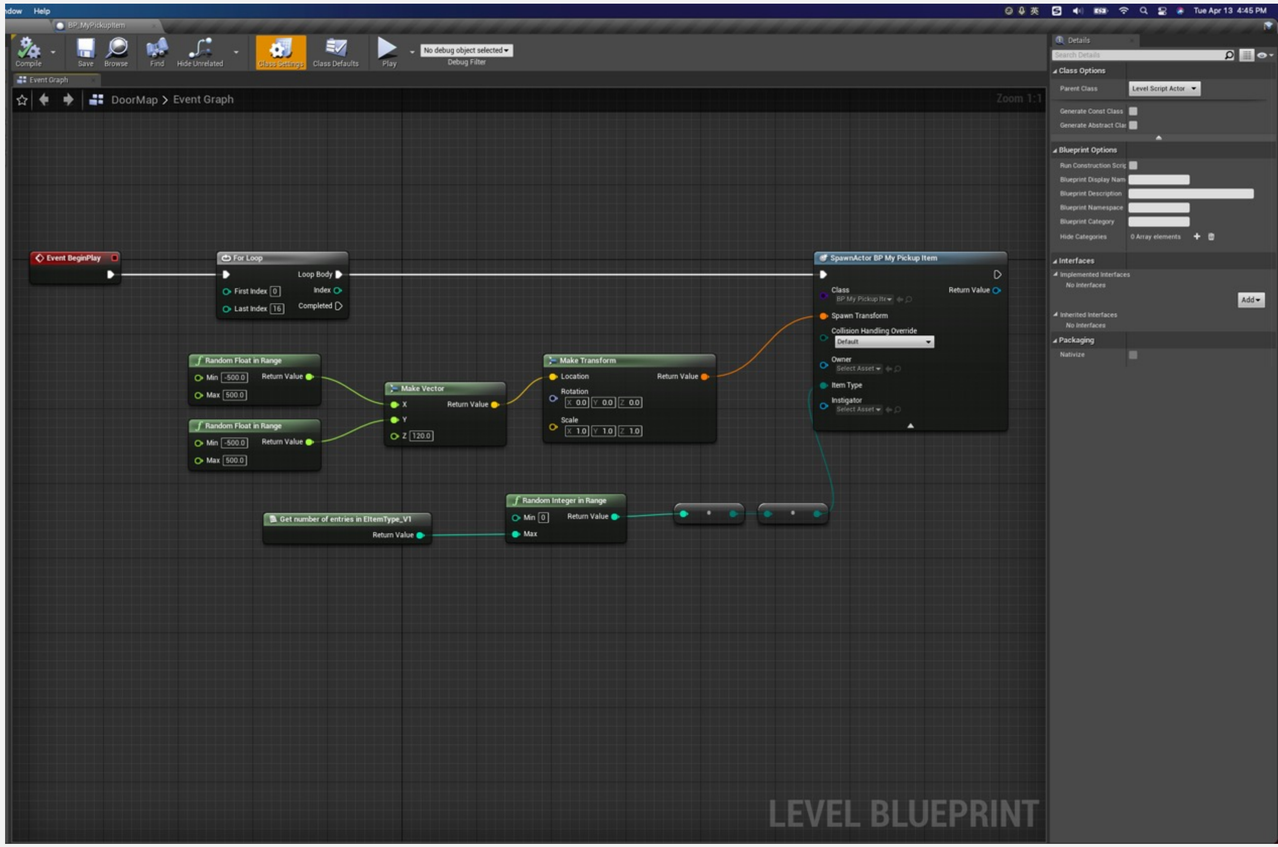
蓝图类(Blueprint Class)

类是对特定类型对象使用的数据和行为的定义。蓝图类可以基于C++类或另一个蓝图类。

蓝图类用于为游戏创建互动对象，并可以在任意关卡中重复使用。

关卡蓝图


Save Current
Source Control
Modes
Content
Marketplace
Settings
Blueprints
Cinematics
Build
Play
Launch
Perspective
Lit
Show
Realtime: Off
Blueprint Class
New Empty Blueprint Class...
Convert Selection to Blueprint Class...
Open Blueprint Class...
Level Blueprints
Open Level Blueprint
Project Settings
GameMode: GameModeBase
World Override
GameMode: Not overridden!
Edit the Level Blueprint for the current level
hold (Cmd + Alt) for more

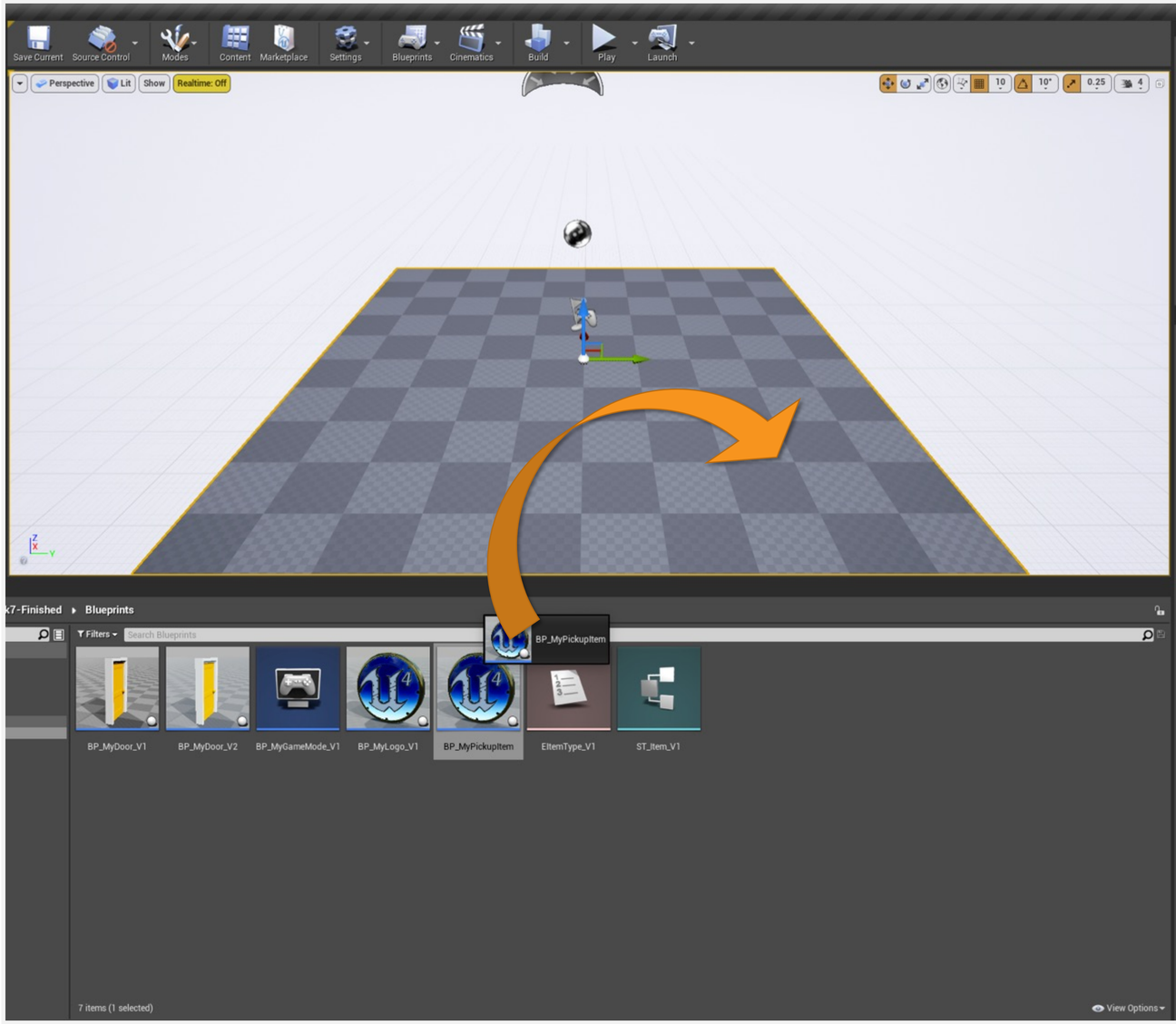

DoorMap > Event Graph
Event BeginPlay
For Loop
Random Float in Range
Make Vector
Make Transform
SpawnActor BP My Pickup Item
Random Integer in Range
LEVEL BLUEPRINT

Actor 实例

Actor Instances

类与实例

- 蓝图类
- 实例化
- 关卡中的对象



类的实例

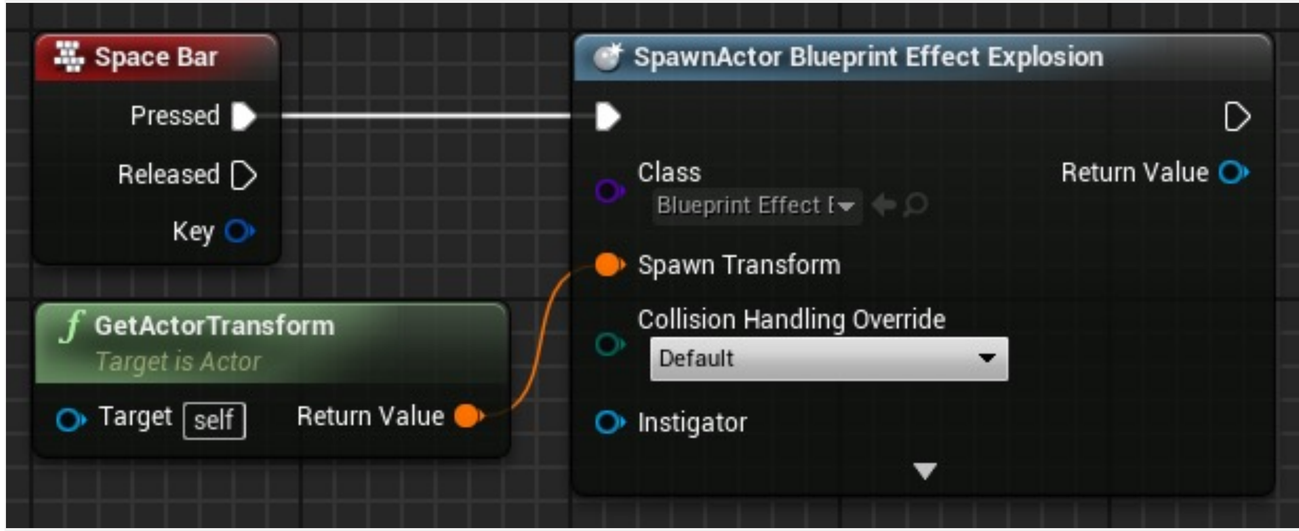
“**实例**”一词用于指代类的对象。

参见右图的示例。假设有一个类名为“Blueprint_Chair”，它表示椅子，右图所示为关卡中添加了Blueprint_Chair类的四个**实例**。

程序生成 Actor 实例

- Spawn Actor from Class 节点：使用指定的类产生 Actor 实例
- 参数：碰撞处理（Collision Handling Override）指定在创建时定义如何处理碰撞。
- 返回值（Return Value）为新创建的实例。

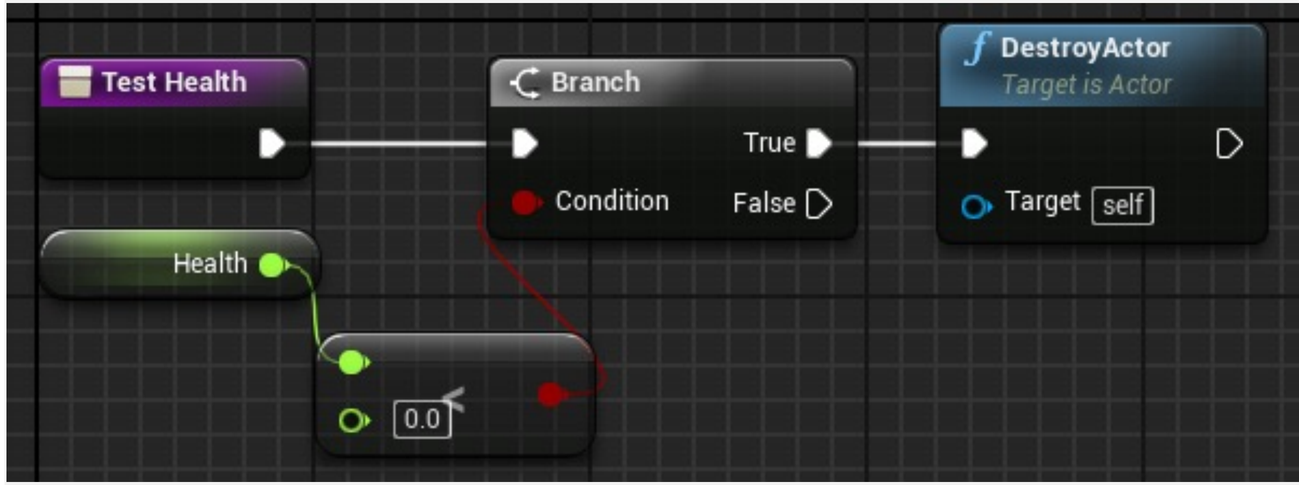
在右图示例中，当按下空格键时，将在当前蓝图的同一个位置创建一个爆炸效果蓝图类（Blueprint Effect Explosion）的实例。



销毁 Actor

- DestroyActor 函数在运行时从关卡中移除Actor实例。
- 要移除的实例必须在目标（Target）参数中指定。

右图显示一个名为“测试生命值”（Test Health）的函数，用于检查“生命值”（Health）变量的值是否低于0。如果为“true”，则由“self”表示的该蓝图的当前实例将被销毁。



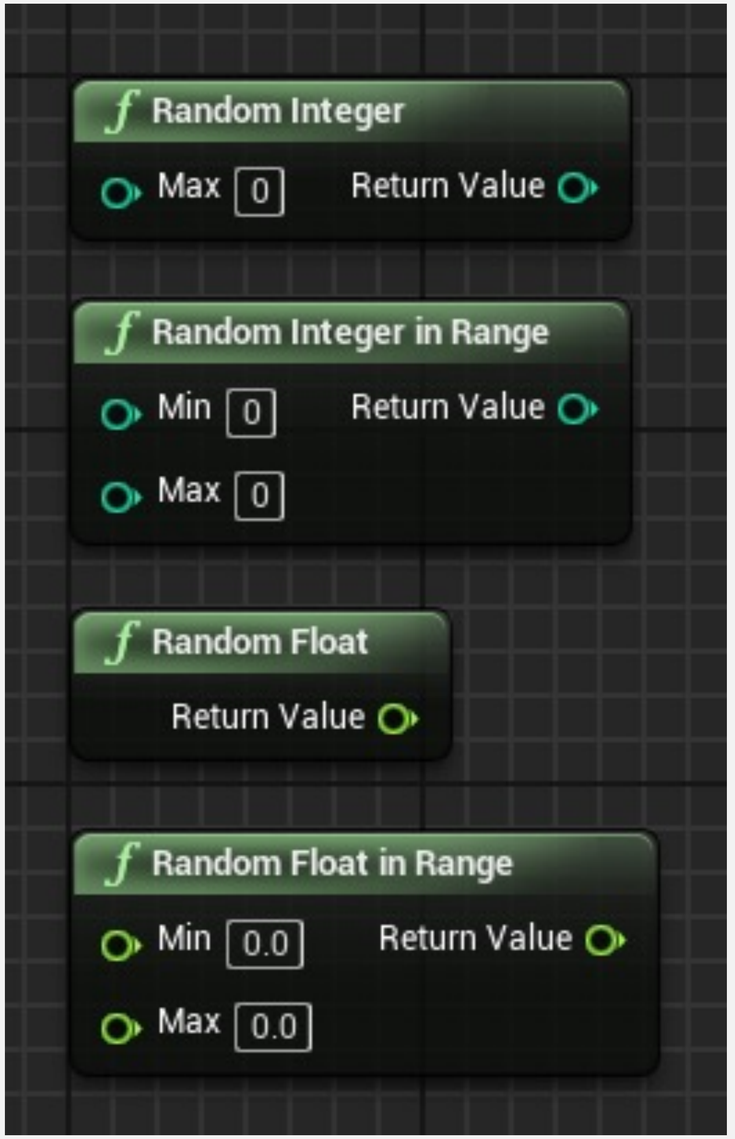
随机数

Random numbers

生成随机数

引擎提供了一些随机数生成函数，主要有：

- 随机整数（Random Integer）：返回介于“0”和“Max”之间的整数值。
- 范围内的随机整数（Random Integer in Range）：返回介于“Min”和“Max”之间的整数值。
- 随机浮点数（Random Float）：返回介于“0-1”之间的浮点值。
- 范围内的随机浮点数（Random Float in Range）：返回介于“Min”和“Max”之间的浮点值。



小 目 标 7

集齐道具解锁

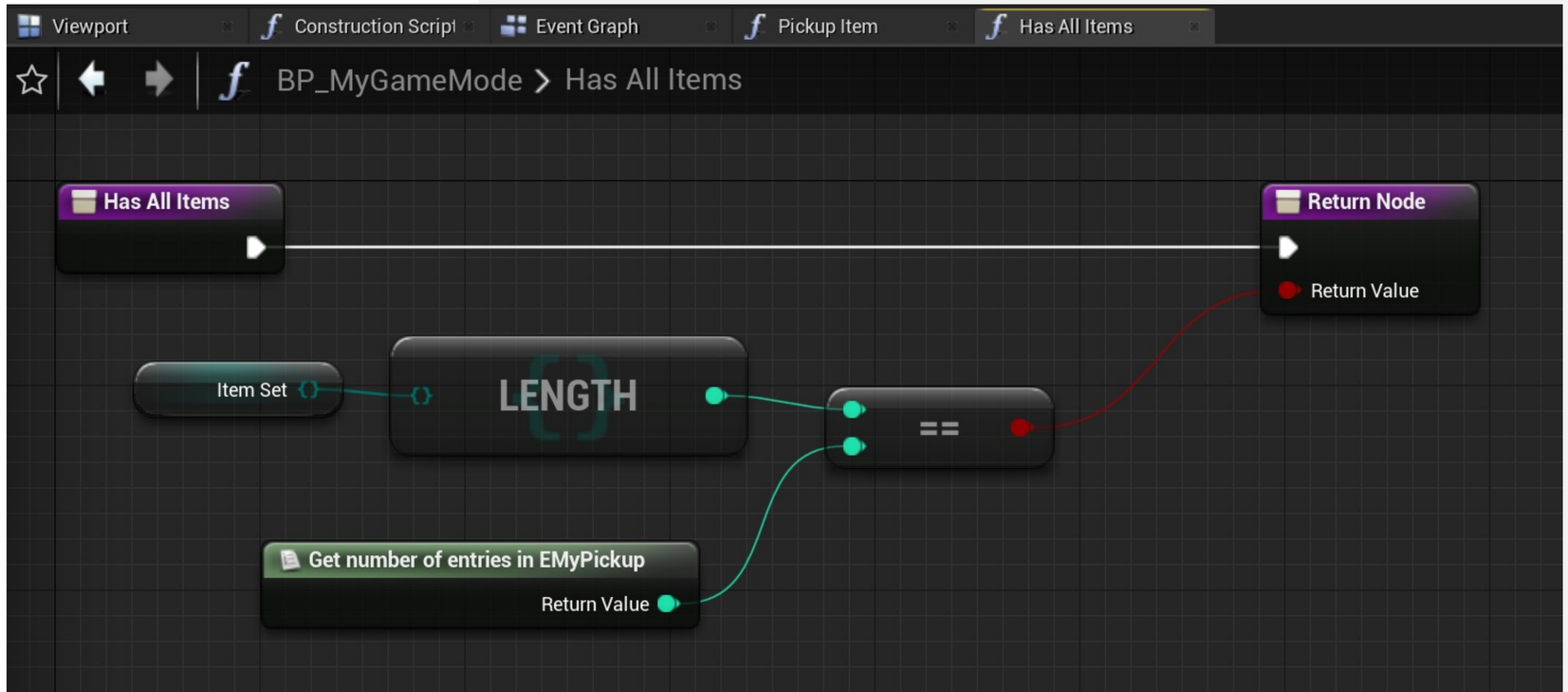
小目标7

对于锁住的门：

- 如果玩家收集齐了所有道具，则自动开锁
- 使用文本格式化，打印更多信息

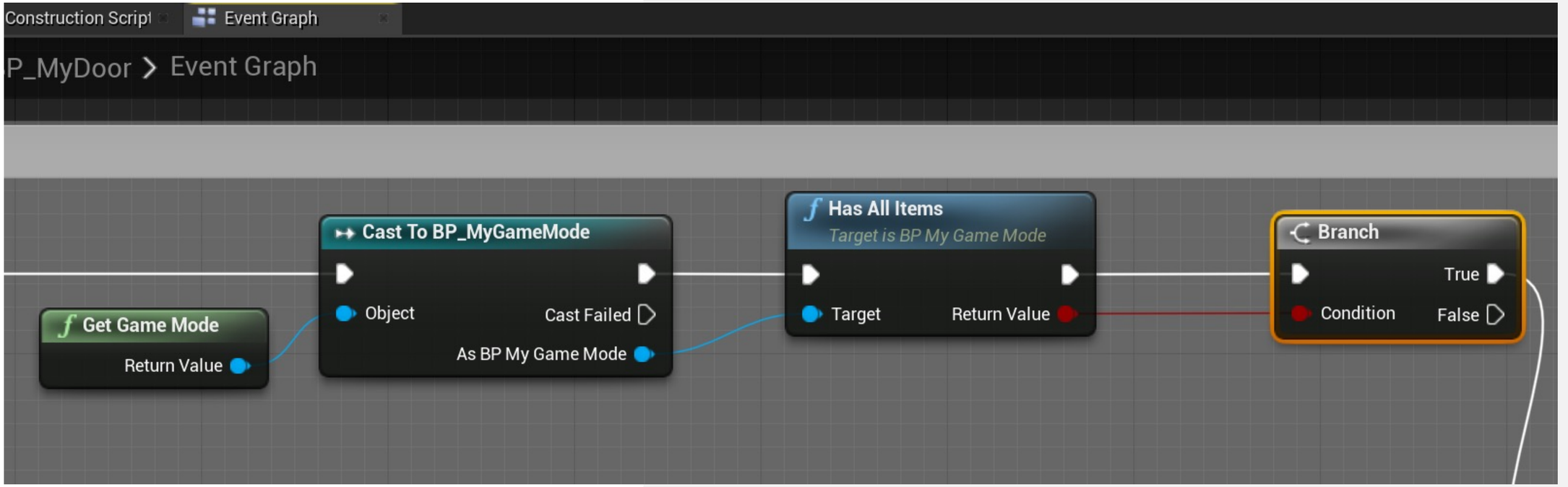
功能实现

- 检测玩家是否收集齐了所有道具类型



功能实现

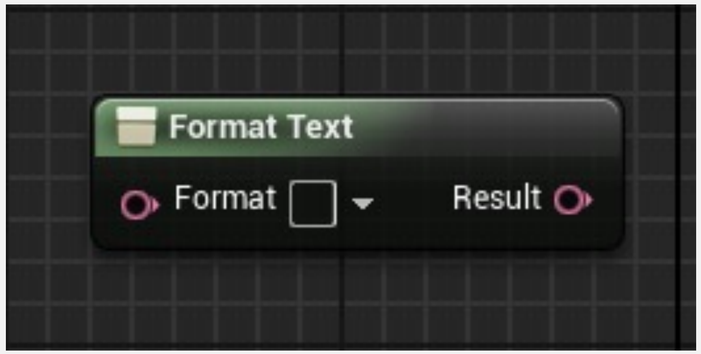
- 自动解锁开门


Construction Script
Event Graph
P_MyDoor > Event Graph
Get Game Mode
Return Value
Cast To BP_MyGameMode
Object
Cast Failed
As BP My Game Mode
Has All Items
Target is BP My Game Mode
Target
Return Value
Branch
True
Condition
False

字符串/文本

String/Text

文本格式化（Format Text）节点



文本格式化（Format Text）节点根据可以在“格式”（Format）参数中指定的参数构建节点的输入针脚，最终输出一个完整的文本。

输入

- Format：接收将成为最终结果一部分的文本。要设置参数，只需为每个参数在定界符{ }中输入名称即可。
- “Format”中定义的参数（Parameters defined in “Format”）：对于每一对花括号，将用花括号括起名称来生成新输入参数。

输出

- 结果（Result）：输出用 Format 参数和其他参数的值构建的最终文本。

文本格式化（Format Text）节点： 示例

在右侧示例中，比赛结束时将出现一个文本，用来显示结果。该文本包含四个变量的值，它们分别表示两位玩家的名称和分数。

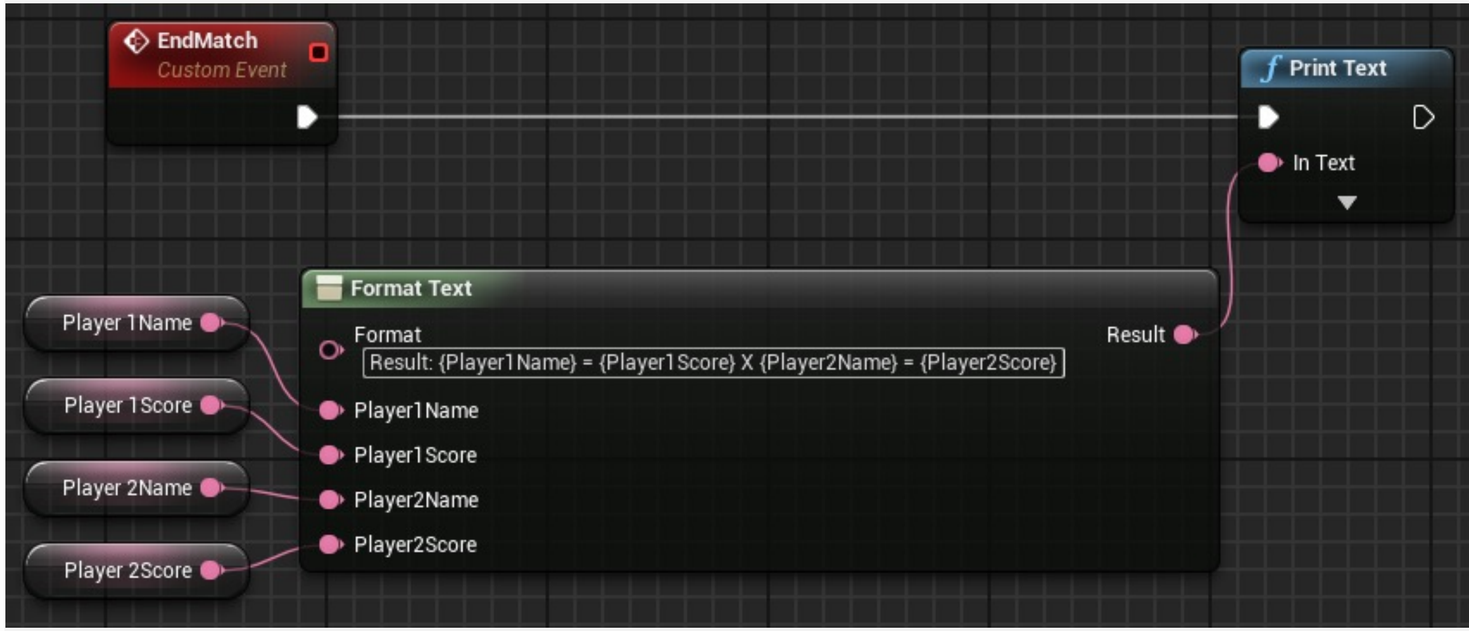
Format 参数中使用的文本如下：

{Player1Name} = {Player1Score} X {Player2Name} = {Player2Score}

放置Format参数的值后，蓝图编辑器生成其他输入参数。

右上图显示了Format Text节点。

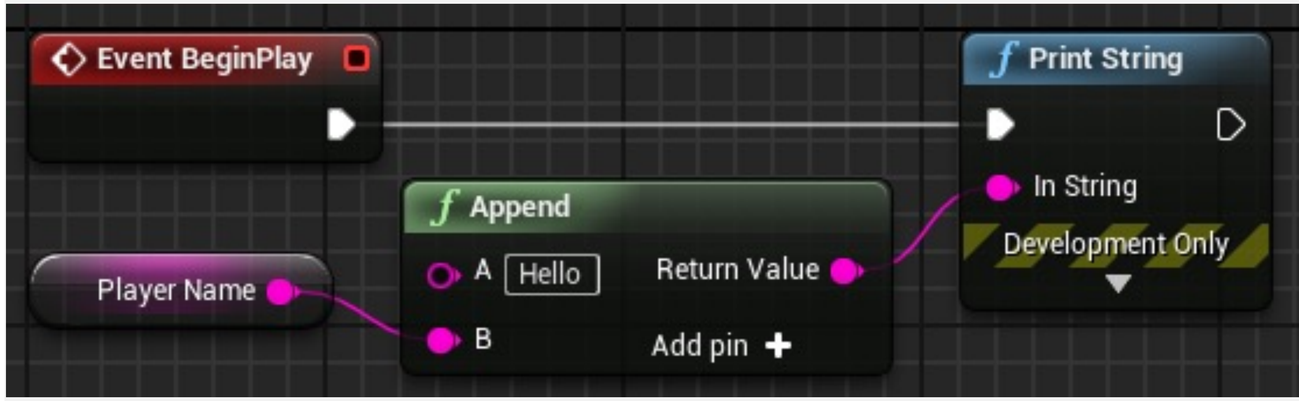
右下图显示了所生成文本的示例。



字符串附加（Append）节点

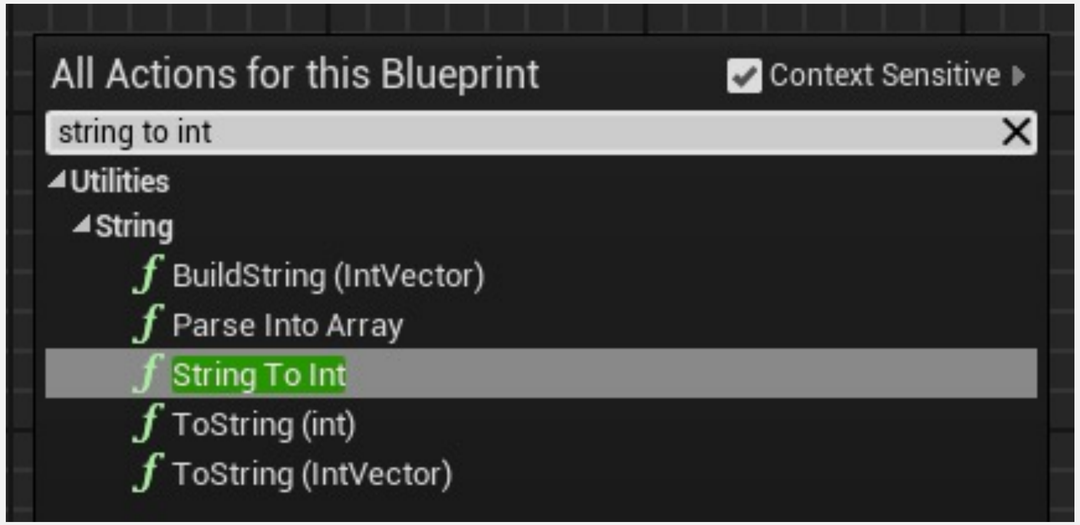
- 附加（Append）节点合并字符串来创建新字符串。
- 可以使用“添加引脚+”（Add pin +）选项添加更多字符串。

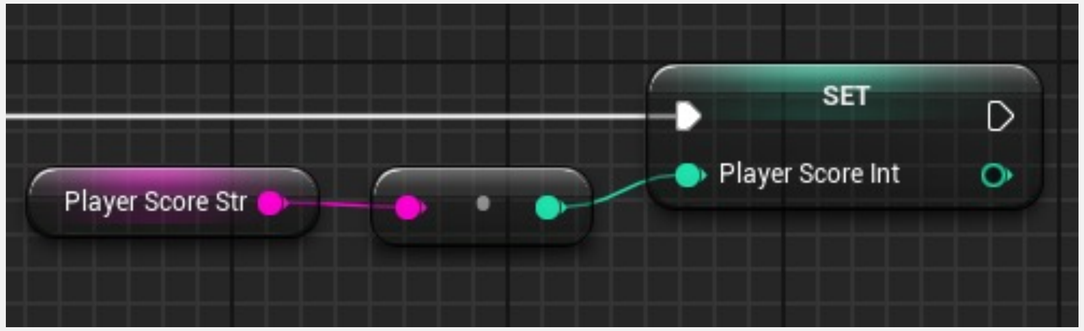
在右侧示例中，将使用“Player Name”创建自定义的欢迎消息（Player Name 是一个字符串变量）。



将字符串解析为数字

- 为了将字符串值转换为整数值，您可以使用函数节点“String To Int”节点 和 “String To Float”节点。
- 如果把“字符串”和“数字”进行数据连线，编辑器将自动创建转换节点，如右下图示例所示。





总结一下

Summary

蓝图作为一种可视化脚本语言

- 开发环境：创建蓝图、蓝图编辑器
- 蓝图是基于事件的
- 基本变量类型
- 操作符、函数
- 控制流程：顺序、分支、循环
- 基本数据结构：数组、Set、Map

谢 谢 大 家 !

Questions?