

## 第 10 课 蓝图节点(9)

### 11、随机 (Random) 类型节点

随机类型的众多节点都属于“数学 (Math)”类目。本节介绍关于产生随机数的随机 (Random) 节点和控制随机性的随机流 (Random Stream) 节点。

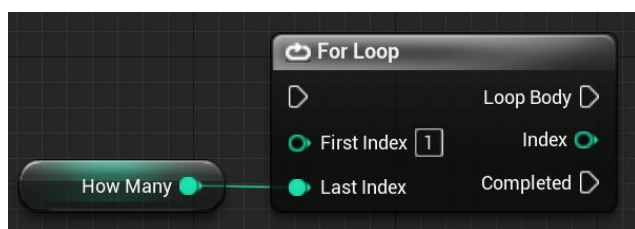
#### (1) Random 随机数

随机节点的用法基本类似，本节讲解最常用的随机单位向量 (Random Unit Vector) 节点，实现功能：在场景中创建一个整型变量用于控制静态网格模型的数量，并在随机位置生成模型。

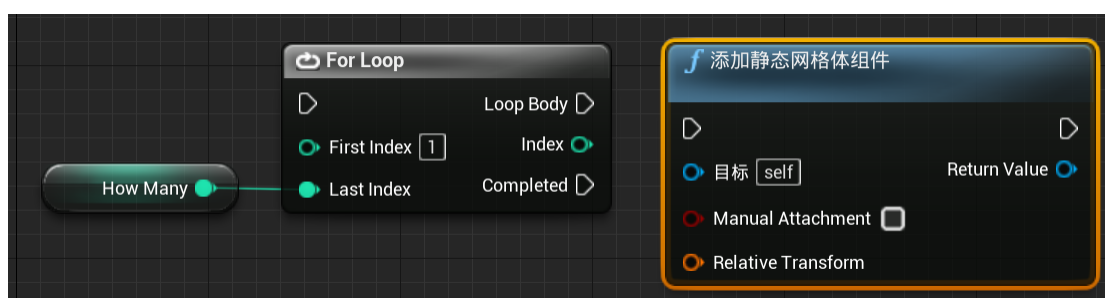
创建一个以 Actor 为父类的蓝图，命名为 ManyBoxes。打开 ManyBoxes 蓝图，切换到构造脚本 (Construction Script) 面板，添加 ForLoop 节点。创建一个整数类型的变量，命名为 HowMany，勾选“可编辑实例”属性，该变量用于设定静态模型的数量。



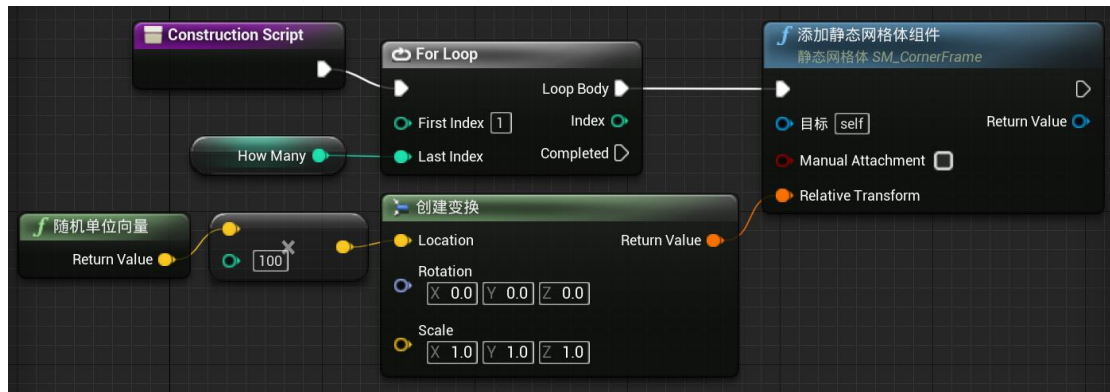
修改 ForLoop 节点的 First Index 为 1，Last Index 为 HowMany 变量的值。



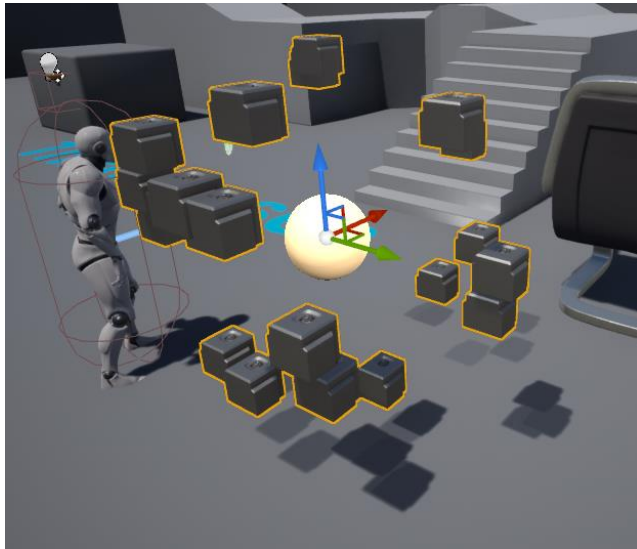
添加“添加静态网格体组件（Add Static Mesh Component）”节点，在节点的细节面板中把静态网格体（Static Mesh）改为 SM\_CornerFrame，也可以把主界面内容浏览器的静态模型拖到节点细节面板的静态网格体中。



在刚添加的“添加静态网格体组件”节点中，有一个输入引脚叫“Relative Transform”，它用来设置添加的静态网格体的相对位置。由于每个静态模型的位置需要随机放置，因此要用到“随机单位向量（Random Unit Vector）”节点。把随机单位向量放大 100 倍作为添加的静态网格体的相对位置。还要用到“创建变换（Make Transform）”节点。



编译保存 ManyBoxes 蓝图，在游戏场景中创建一个 ManyBoxes 对象，修改对象细节面板中 How Many 选项的值为 21。

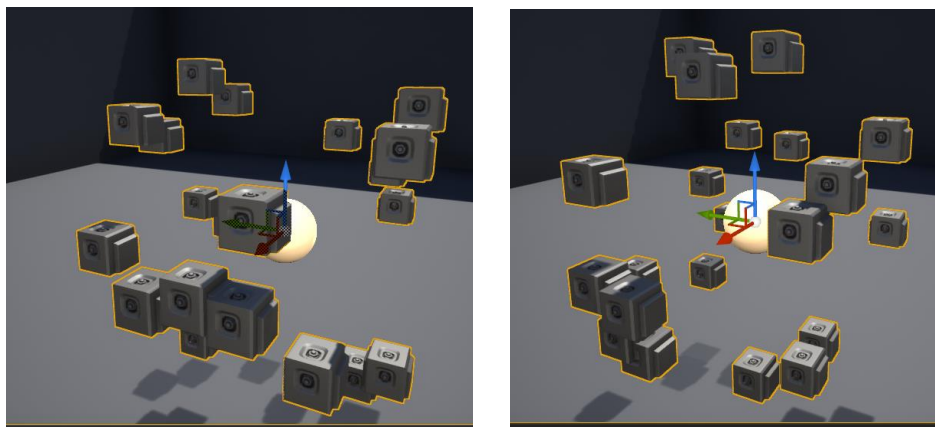


运行游戏。



## (2) 随机流 (Random Stream)

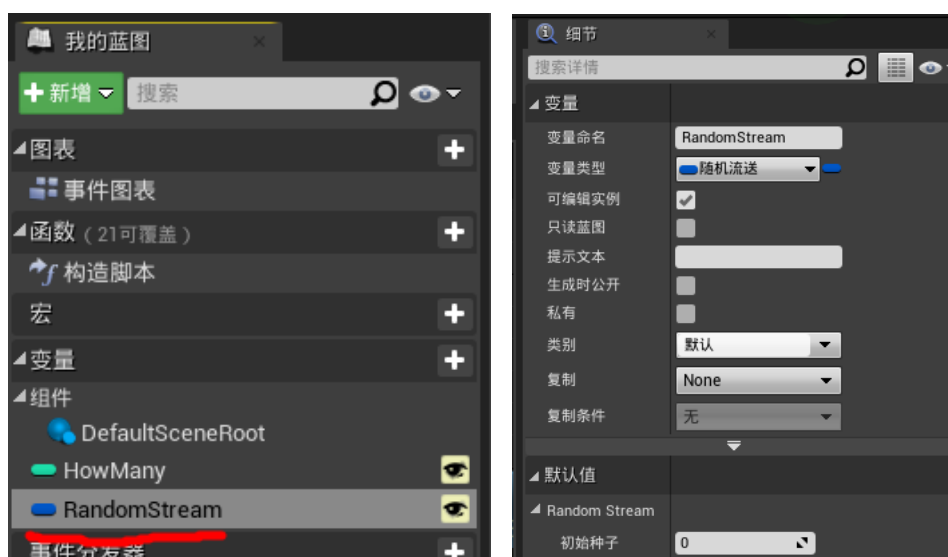
在上一节定义的 ManyBoxes 对象定义了静态模型的数量且随机分布，但可以发现，当在场景中移动 ManyBoxes 对象时，所有随机分布的静态模型会重新随机定位。



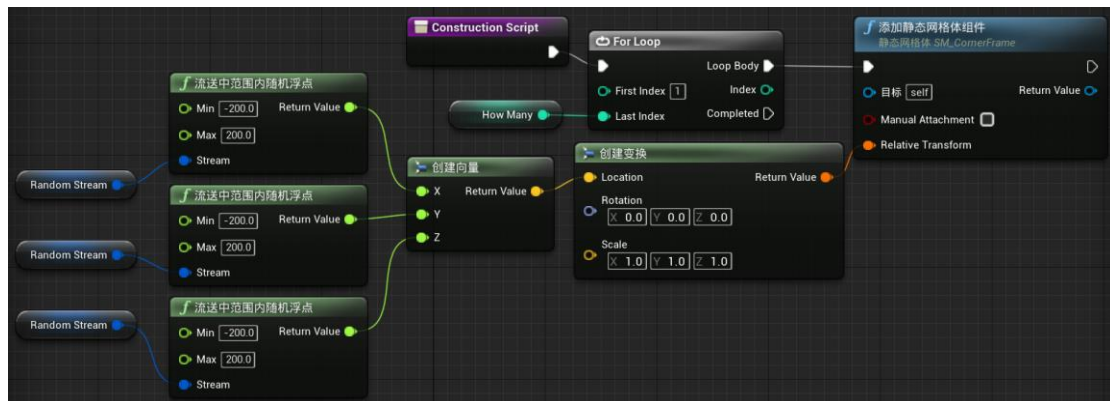
这是因为在游戏未运行时对 ManyBoxes 进行的任何操作都会执行该蓝图的构造脚本，因此随机单位向量节点会重新为每个静态模型生成新的随机坐标。

有时候并不希望改变对象位置时重新生成随机值，那么就要利用随机流。随机流允许在蓝图、关卡蓝图及针对动画的动画蓝图中重复生成相同的随机数。

打开 ManyBoxes 蓝图，添加一个随机流送 (Random Stream) 类的变量，命名为 RandomStream，勾选“可编辑实例”选项。



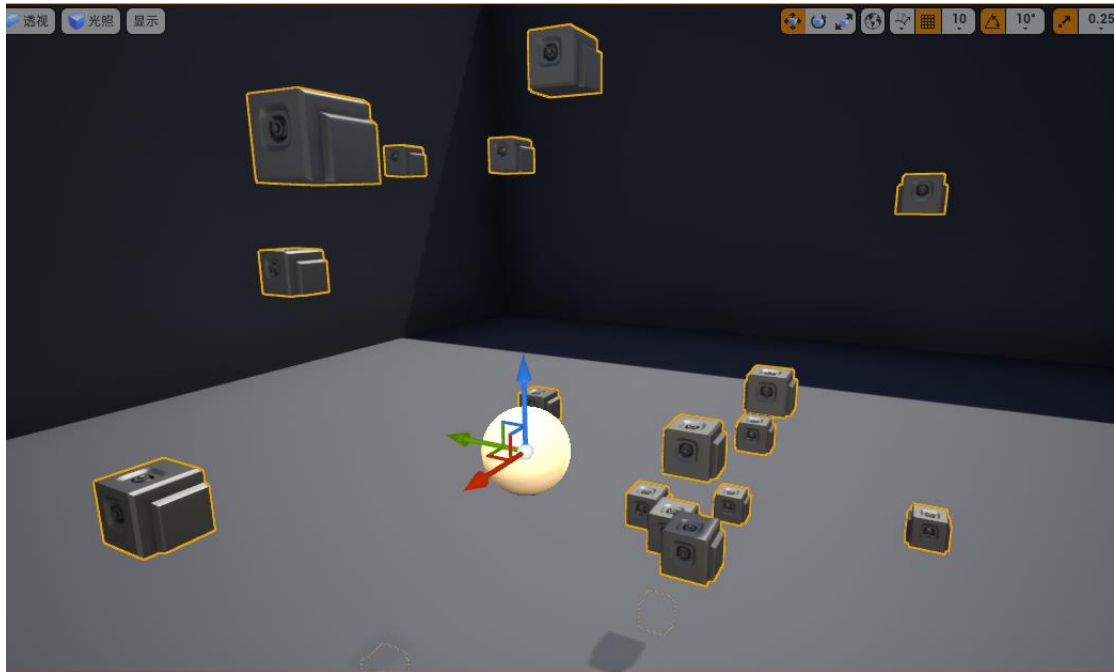
打开 ManyBoxes 蓝图的构造脚本编辑页面。删除“随机单位向量”节点和“向量\*整数”节点。把 RandomStream 变量拖入蓝图中，复制三份。创建“流送中范围内随机浮点（Random Float in Range from Stream）”节点，设置 Min 和 Max 的值分别为-200 和 200，复制三份。创建“创建向量（Make Vector）”节点。



编译并保存 ManyBoxes 蓝图，返回主界面，在 ManyBoxes 对象的细节面板中可以编辑 RandomStream 变量的初始种子。可以看到每当初始种子发生变化时，静态模型的分布会进行改变。



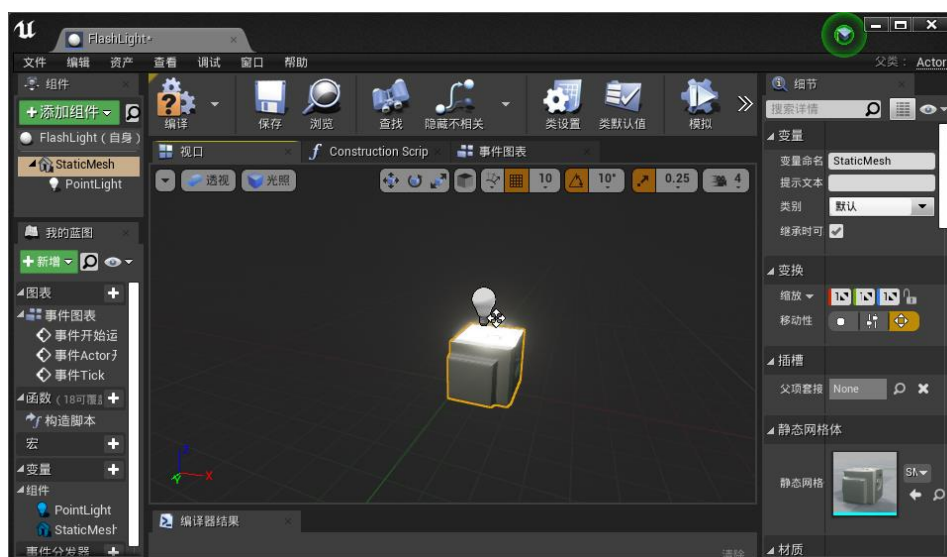
当初始种子值保持不变时，在游戏场景中移动 ManyBoxes 对象，可以看到所有静态模型的分布在移动过程中没有发生变化。



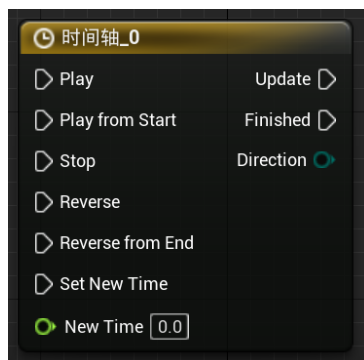
## 12、时间轴（Timeline）节点

时间轴节点是蓝图中的特殊节点，该节点用于快速设计基于时间的简单动画，并基于游戏中的事件进行播放。时间轴节点专门用来处理简单的、非过场动画的任务，可以随着动画的播放触发事件。本节通过实现灯光的闪烁效果来学习时间轴节点的法。

在主界面的内容浏览器中找到 SM\_CornerFrame 静态模型，并以它为组件创建蓝图，命名为 FlashLight。添加“点光源（PointLight）”组件。



在 FlashLight 的事件图表面板，添加“添加时间轴（Add Timeline）”节点。节点默认名字为“时间轴\_0”。



该节点左侧有 6 个执行引脚和 1 个数据引脚。6 个执行引脚用来触发该时间轴的播放模式，分别如下：

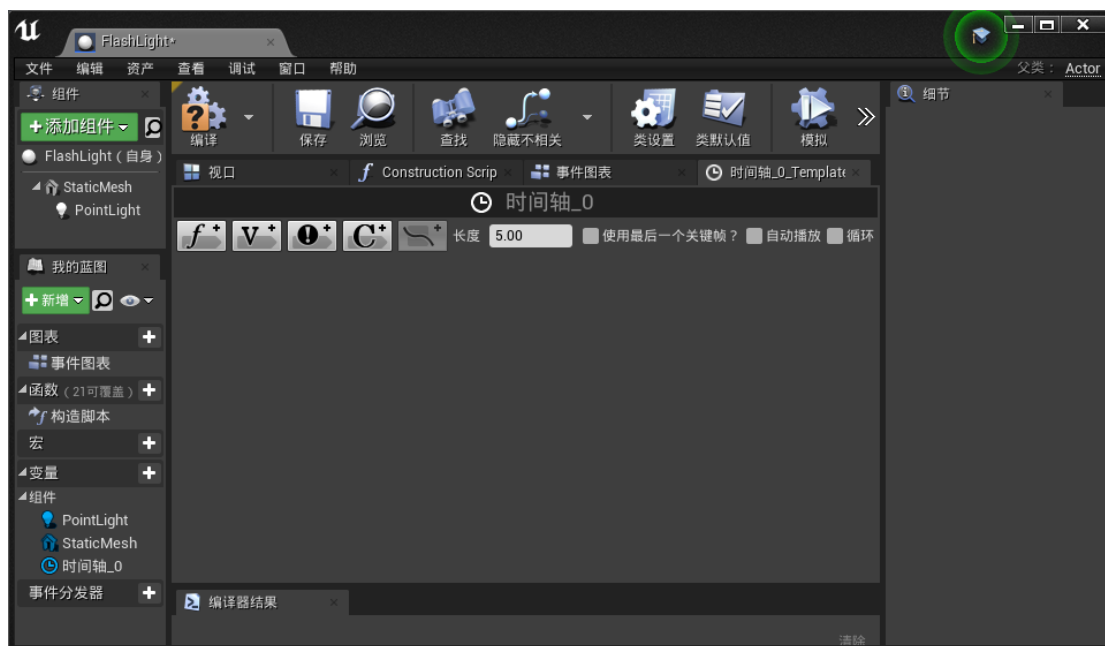
- (a) Play（播放）：时间轴从当前时间处开始正向播放
- (b) Play from Start（从头播放）：时间轴从开始处正向播放
- (c) Stop（暂停）：在当前时间处停止播放时间轴
- (d) Reverse（反向播放）：从当前时间处反向播放时间轴
- (e) Reverse from End（从尾反向播放）：从结尾处反向播放时间轴
- (f) Set New Time（设置新时间）：将“New Time”值设置为当前时间

时间轴节点右侧有 2 个执行引脚和 1 个数据引脚，分别如下：

- (g) Update（更新）：每当调用当前时间轴就输出一个执行信号
- (h) Finished（完成）：当播放结束时输出一个执行信号，该引脚不会被输入引脚 Stop 触发
- (i) Direction（方向）：输出枚举数据，指明时间轴的当前播放方向（快进/快退）

双击“时间轴\_0”节点，会进入时间轴编辑器（Timeline Editor）。在这个面板中可以对时间轴进行编辑。





编辑器中各按钮或复选框的功能：



**f<sup>+</sup>**：添加浮点型轨道到时间轴，对浮点值进行动画处理。

**V<sup>+</sup>**：添加向量型轨道到时间轴，对向量值（如旋转值、平移值）进行动画处理。

**!**：添加事件轨道到时间轴，该轨迹会提供另一个执行输出引脚，此引脚将在轨迹的关键帧时间处被触发。

**C<sup>+</sup>**：添加颜色轨道到时间轴，对颜色进行动画处理。

**曲线**：添加选中的曲线资产到时间轴，此按钮仅当在内容浏览器中选择外部曲线后才能被激活。

长度 (Length)：设置时间轴长度，单位秒。

使用最后一个关键帧？ (Use Last Keyframe?)：如果不勾选此选项，将忽略序列的最后关键帧。如果勾选此选项，表示使用时间轴最后一个关键帧所在的时间点作为结束时间，而不是使用设置的长度 (Length) 作为结束时间点。




自动播放 (AutoPlay): 如果勾选, 时间轴动画无须输入信号即可在关卡开始时播放。

循环 (Loop): 如果勾选, 时间轴动画会无限循环播放, 除非通过 Stop 输入引脚停止。

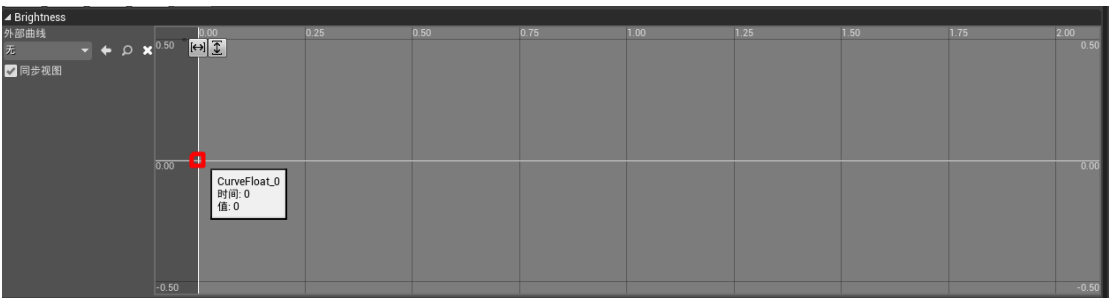
已复制 (Replicated): 如果勾选, 时间轴动画将跨客户端被复制。

忽略时间膨胀 (Ignore Time Dilation):

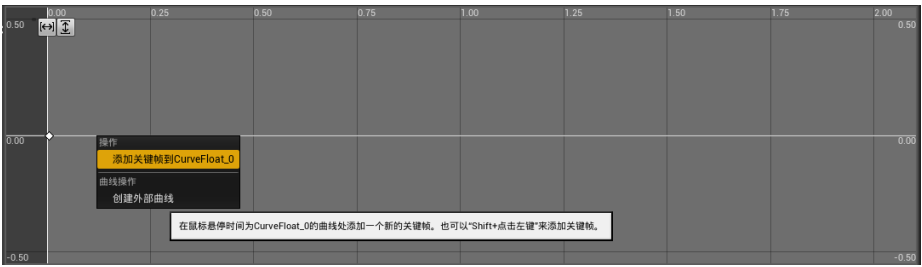
点击  添加浮点型轨道按钮, 由此新建了一个轨道“新建轨道\_0”, 界面布局如图:



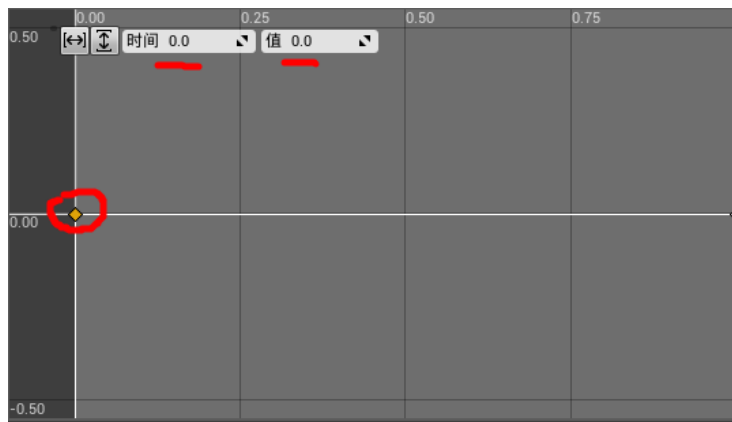
把轨道名称改为 Brightness, 然后在时间为 0 处的函数线 CurveFloat\_0 上按住 Shift 键并单击鼠标左键添加关键帧。



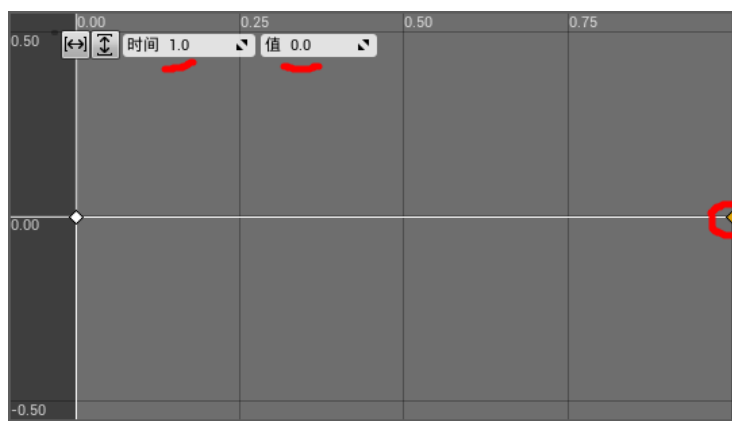
也可以单击鼠标右键, 选择“添加关键帧到 CurveFloat\_0”。



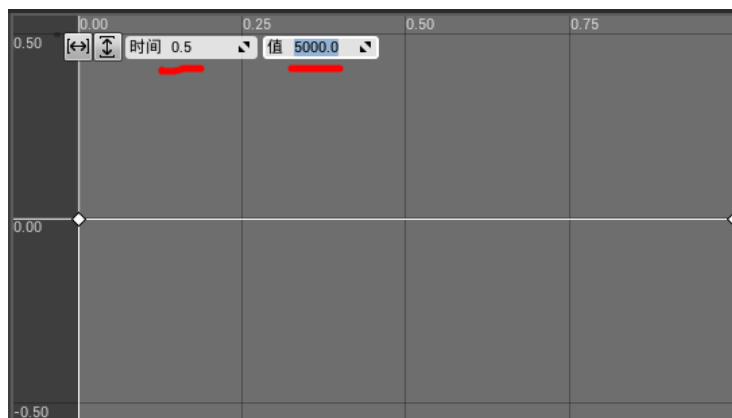
把 Timeline\_0 的长度修改为 1，并添加 3 个关键帧。选中其中一个关键帧，调整“时间”和“值”均为 0。




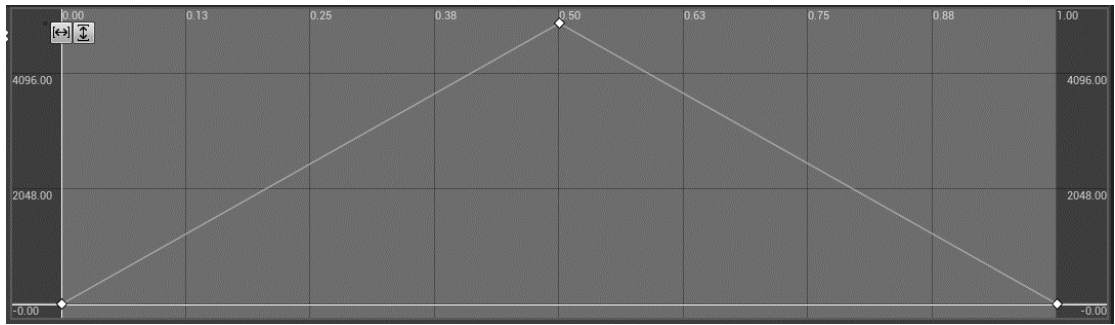
选中第二个关键帧，把“时间”设置为 1， “值”设置为 0。



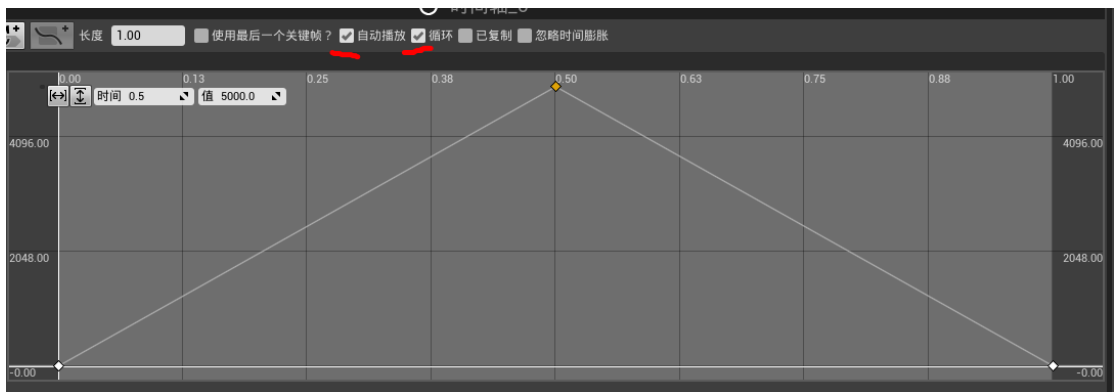
选中第三个关键帧，把“时间”设置为 0.5，“值”设置为 5000。



由于第三个帧的值很大，超出默认范围 $\pm 0.5$ ，如果在横纵范围内想要看全，可以按  键，缩放到合适的比例范围。

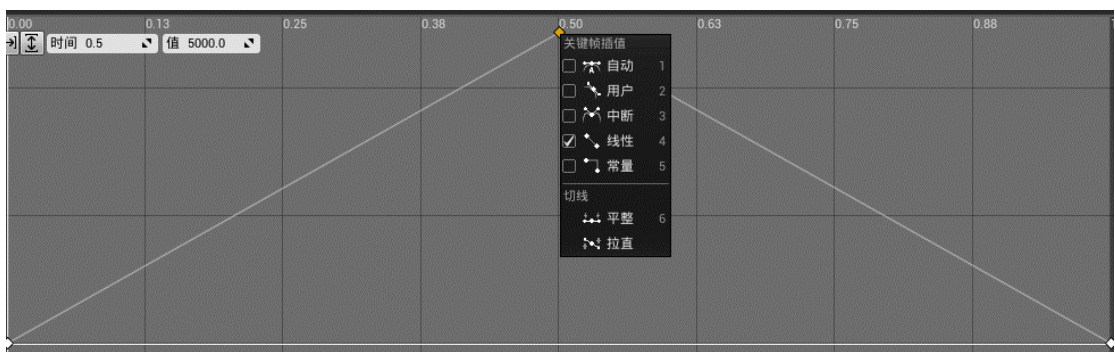


最后勾选“自动播放 (AutoPlay)”和“循环 (Loop)”。

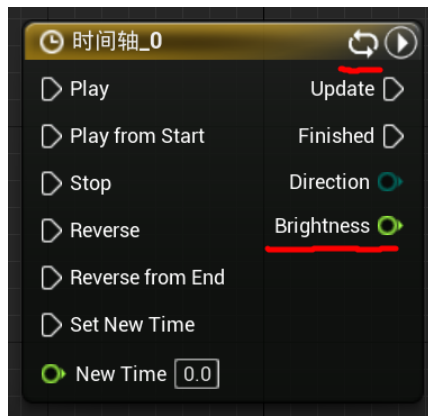


如果想删除关键帧，可以通过选中关键帧然后按 Delete 键。可以通过拖动关键帧改变时间和值的信息，但最好直接编辑会比较精确。

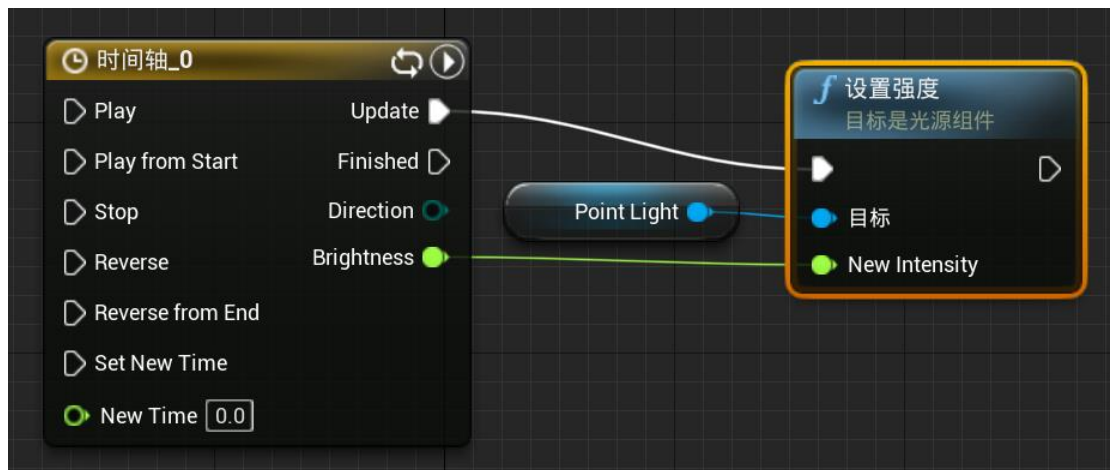
通过鼠标右键点击关键帧，还可以选择关键帧的插值类型，如同曲线插值一样。当前曲线默认插值类型为线性 (Linear)。



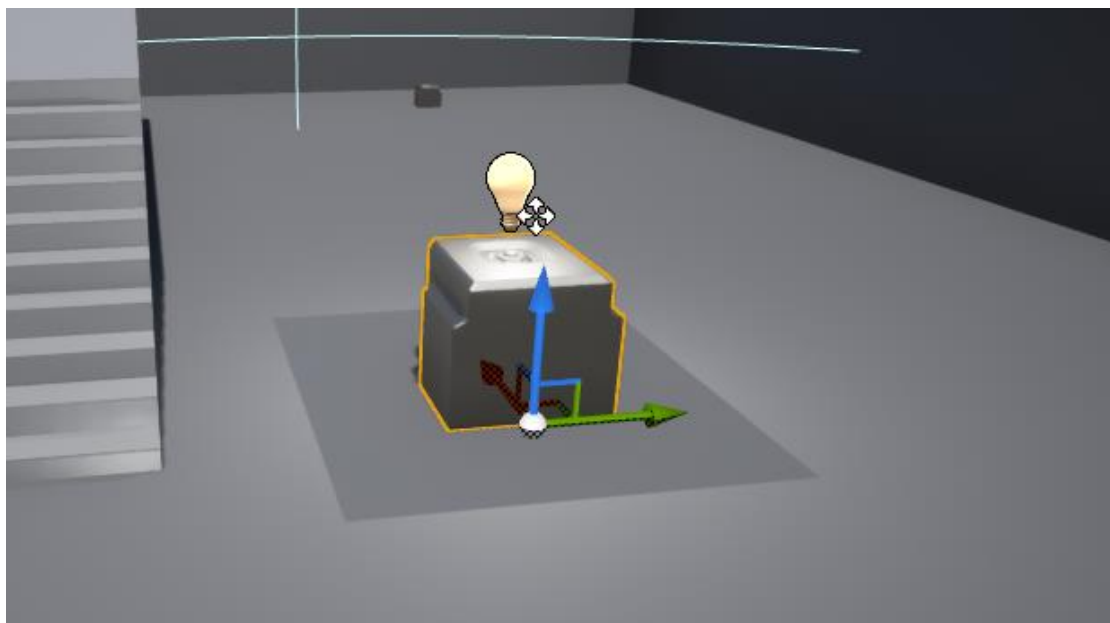
完成编辑轨迹后，定义轨迹的数据或事件执行将由与轨迹名称相同的数据或执行引脚来输出。编译保存，返回事件图表面板，可以看到 Timeline\_0 节点右上方与一个循环标志出现，且右侧增加了一个 Float 类型的返回值 Brightness，该返回值返回的时 Brightness 轨道中根据 CurveFloat\_0 曲线与时间相对应的值。



在事件图表中获取对 PointLight 组件的引用，利用它找到“设置强度（Set Intensity）”节点，用于调节点光源的灯光亮度。

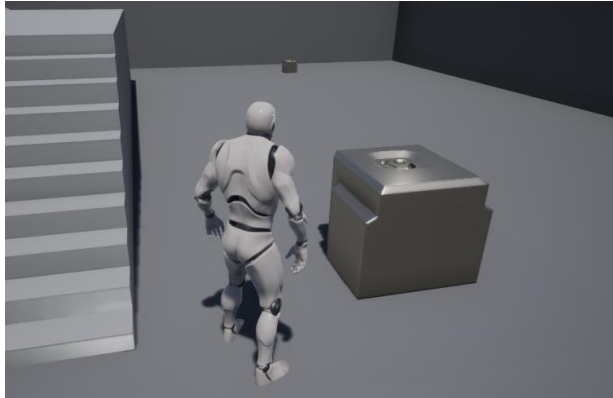


把 FlashLight 对象放置在游戏场景中。



运行游戏可以看到，灯光强度每一秒钟就会进行一次从暗到亮再到暗的周期

变化。



作业：建立 4 个点光源，假设称为 1~4 号。在游戏开始运行后，即开始按以下规律变换：第 1 秒亮 1 号，第 2 秒亮 2 号，第 3 秒亮 3 号，第 4 秒亮 4 号，然后重复，第 5 秒亮 1 号，…。使用两种方法实现：1、所有点光源定义为同一种类型，用“获取类的所有 Actor（Get All Actor of Class）”节点实现；2、使用本节介绍的 Timeline 实现。比较两种实现方法各自擅长使用的场景。