## AI路径巡逻

AI寻路我们通过给他增加巡路点来实现。同时可以让AI根据寻路点的位置进行周围游荡。

创建Actor类蓝图，命名为BP\_定位点。加上静态网格体组件和设置材质即可。然后修改碰撞为NoCollision。

为了让AI能够找到定位点，我们用组件来做。这样做的好处是可以灵活配置。

创建ActorComponent类蓝图，命名为AC\_AI路径组件。

创建变量，命名为“路径点”，类型为BP\_定位点的映射，值类型为float。我们将值表示为AI在该点的停留时间，默认值为0。

创建变量，命名为“巡逻方式”，类型为E\_巡逻方式。巡逻方式有三种：循环、往返、终止。

上面的两个变量都是可编辑实例，没有生成时公开。

接下来实现路径的选择。在场景中添加定位点和敌人模板，在敌人模板中添加AC\_AI路径组件。在场景中点击AI，找到细节中的该组件，在细节中给路径点赋值。



创建行为树，命名为“行为树模板”。新建任务，继承自BTTask\_BlueprintBase，命名为Task\_按照既定路径巡逻。

重载函数“接收执行AI”，这里ControlledPawn即为控制的AI。我们从AI中Get Component By Class，这样就能设计路径点的赋值。

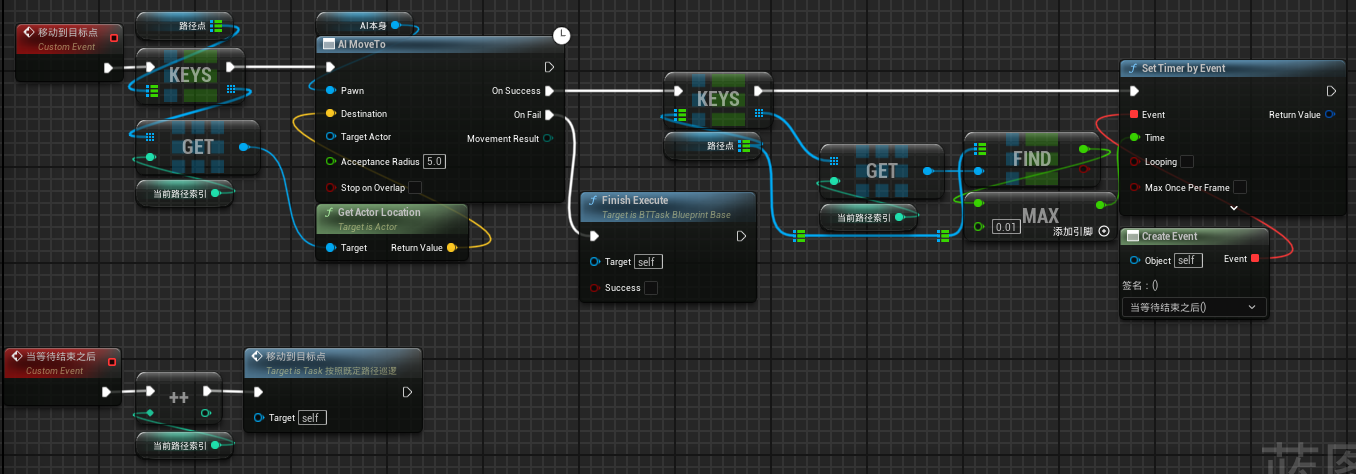
创建变量，命名为“当前路径点索引”，来记录。

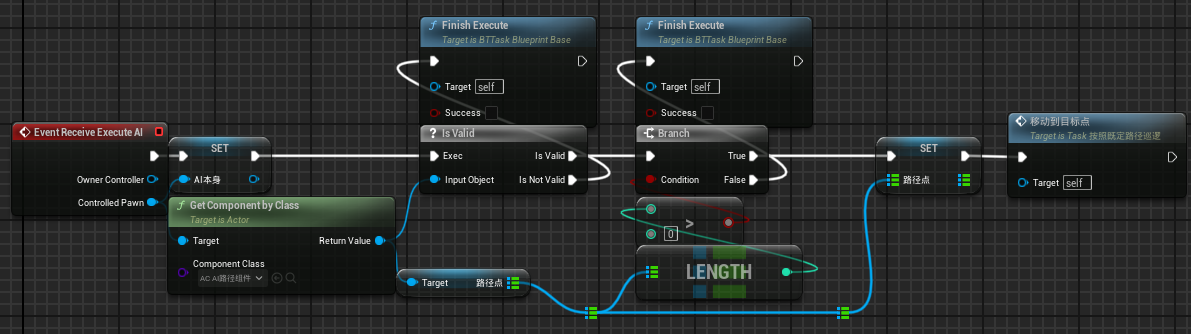
创建自定义事件，命名为“移动到目标点”。



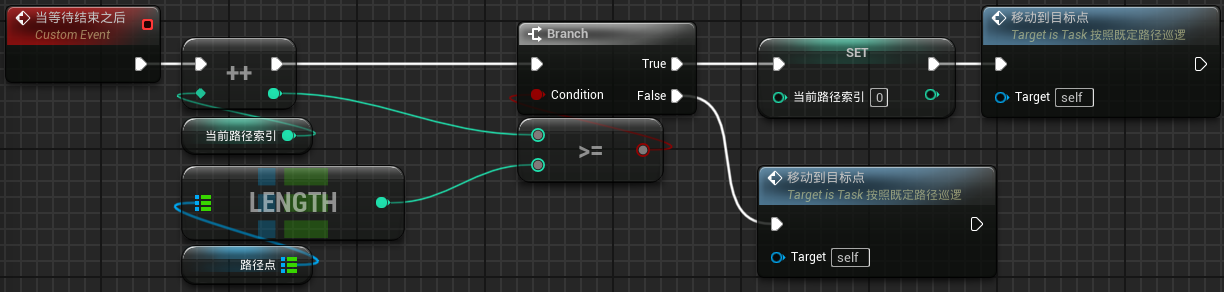
这里GET（a copy）的返回值即为场景中定位点的实例。这时我们拿到他就可以获取到他的位置，就可以让AI走过去了。

为了实现当AI移动到目标点经过values值的时间后再继续移动，我们用SetTimerByEvent。注意定时器的时间不能为0，所以选择Find值和0.01中的最大值。





这时，索引值可能会越界，所以要在索引值++的时候加一个判断



这样就实现了循环的巡逻

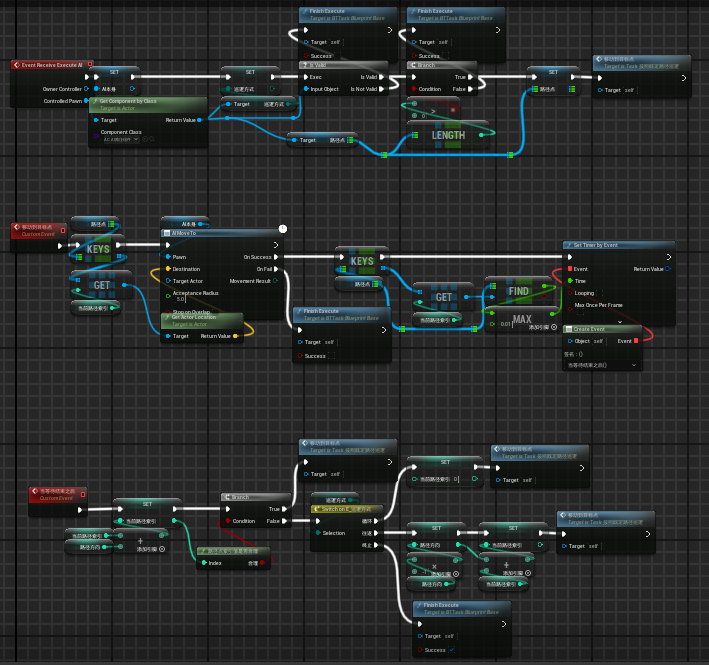
接下来实现另外两种巡逻：往返、终止

问题的关键是定位点索引值怎么变。当往返的时候，索引值是-1，而不是+1。这里我们不用+-1来实现，我们用方向正负来实现。

创建int型变量，命名为“路径方向”，默认值为1。这时，我们需要修改“当等待结束之后”对当前路径索引加上路径方向。

同时我们也需要判断路径点中的某个索引值是否合理，在事件图表中创建下面三个节点，折叠为函数，命名为“路径点索引值是否合理”，纯函数。





在使用该功能的时候，选择场景中的AI选择AC\_AI路径组件在细节中选择需要的寻路方式和定位点即可。

## 在指定位置周围around

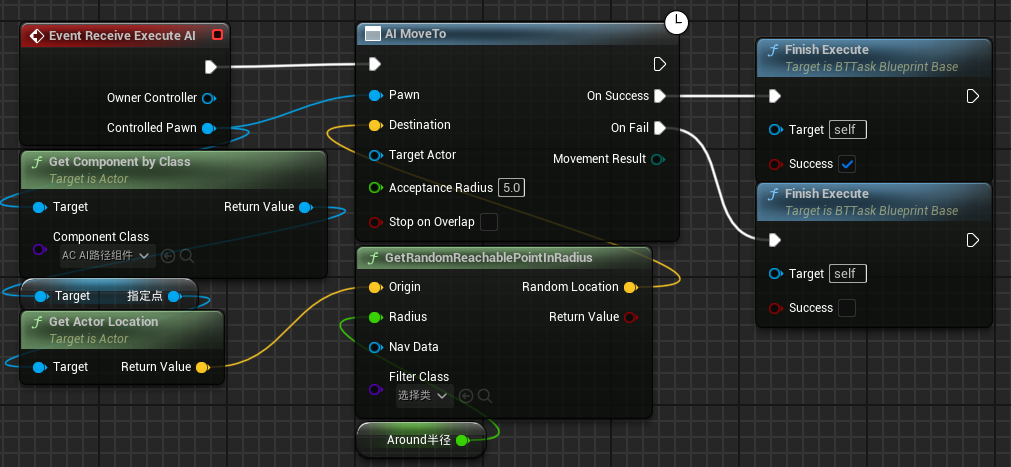
在AC-AI路径组件中创建变量，命名为“指定点”，类型为BP-定位点的对象引用，勾选可编辑实例，不勾选生成时公开

现在在场景中选择目标AI，在细节中找到他的AC-AI路径组件，在场景中选择，在细节中找到默认，给指定点赋值

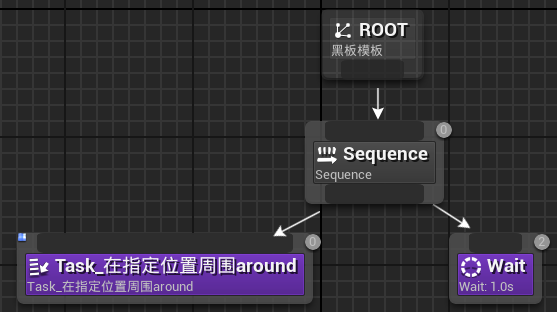


选择右边的吸管，然后在场景中选择目标定位点即可。

然后实现Task\_在指定位置周围around



然后在行为树中简单实现一下即可



# AI行为树与AI视觉感知

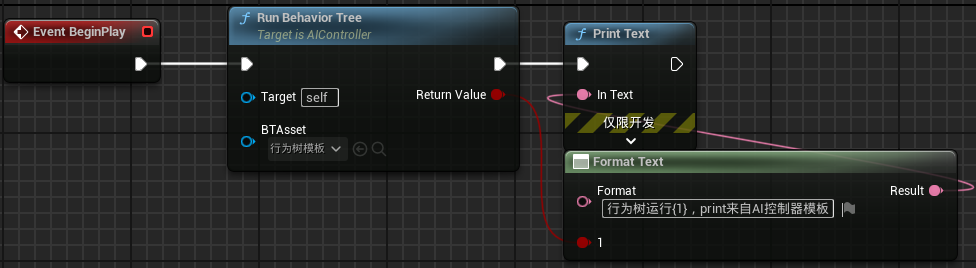
## AI运行逻辑

AI的AI控制器选择指定的AI控制器，由AI控制器运行行为树，在行为树中选择指定的黑板

首先创建AIController蓝图，命名为AI控制器模板。进入BP-敌人模板，选择对应的AI控制器。

创建行为树和黑板，命名为行为树模板、黑板模板。打开行为树，选择对应的黑板。

进入AI控制器模板，当EventBeginPlay时运行行为树。



## AI感知

在AI控制器中添加AI感知组件，感官配置添加视力配置

（一般来说，Controller是一个角色的大脑，角色的感知这类东西应当放到大脑里，而不是身体里，所以将感知组件放到Controller中。另外一种解释是，如果我们希望怪物也能由玩家控制，假设此时将ai感知放到怪物身上，怪物就会多很多不必要的东西，甚至有行为冲突）

AI感知组件默认只能感知到角色类的Actor，如果我们想让他感知到非角色类的物体，需要AI感知源刺激组件

选中AI感知组件，在细节中找到“AI感知--感官配置”，添加视力配置。之后展开它。

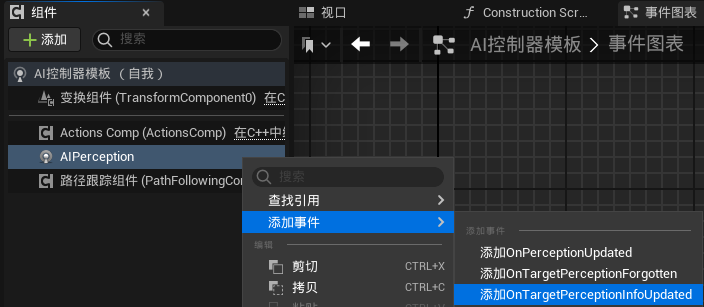


周边视觉半角为视角，给到50

按归属检测全部勾选

注意实现的AISense\_Sight

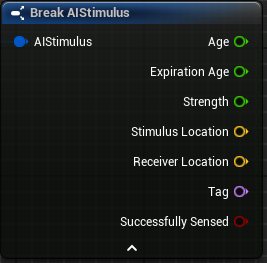
右键AI感知组件，选择添加OnTargetPerceptiopnUpdated事件





Actor是看到的目标

关注它的Stimulus，有如下属性



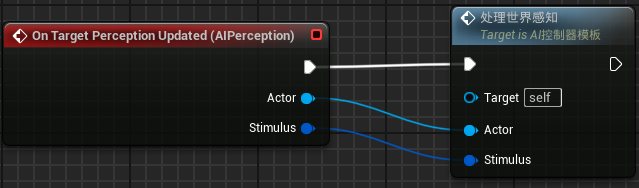
我们将其中的一部分写入到黑板中。

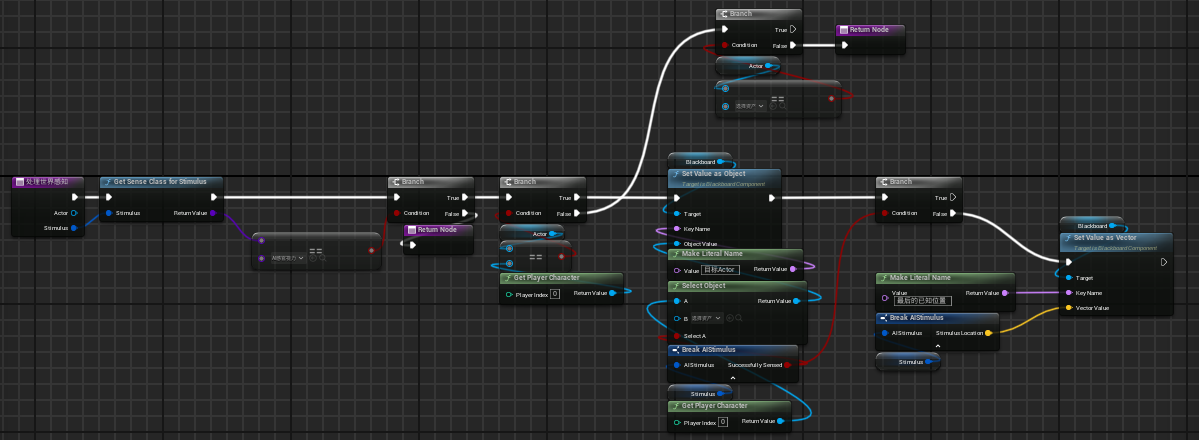
黑板：

目标Actor：键类型Object，基类Actor

最后的已知位置：键类型向量（Stimulus Location）

然后我们就要在控制器中给黑板中这些关键帧赋值。因为这些都是视觉感知，所以我们创建函数命名为“处理视觉感知”





之后就可以进入游戏，开始游戏摁 ’ 键看AI事件

也可以进行为树添加MoveTo任务节点看现象

## 视觉感知

打开AI控制器，添加AI感知组件

（一般来说，Controller是一个角色的大脑，角色的感知这类东西应当放到大脑里，而不是身体里，所以将感知组件放到Controller中。另外一种解释是，如果我们希望怪物也能由玩家控制，假设此时将ai感知放到怪物身上，怪物就会多很多不必要的东西，甚至有行为冲突）

AI感知组件默认只能感知到角色类的Actor，如果我们想让他感知到非角色类的物体，需要AI感知源刺激组件

选中AI感知组件，在细节中找到“AI感知--感官配置”，添加视力配置。之后展开它。



周边视觉半角为视角，给到50

按归属检测全部勾选

注意实现的AISense\_Sight

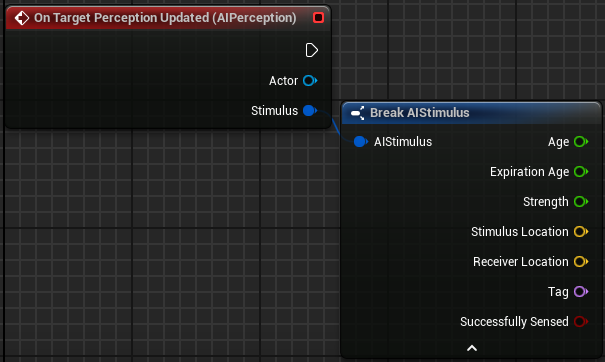
到上个已见位置的自动成功范围：如果此时AI在追着我们跑，我们在一堵墙那里转弯使得AI看不到我们。如果这个范围值为一个正数，那么AI会在这个正数范围内继续追我们。

视点向后偏移&接近剔除半径：两个同时使用。把视锥从眼睛的位置向后偏移，接近剔除。

最大年龄：为0表示不会消失，

现在添加AI看到我们的时候要做的事情

在怪物AI控制器中右键AI感知组件，选择“添加事件--目标感知更新时”

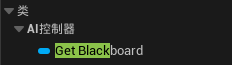


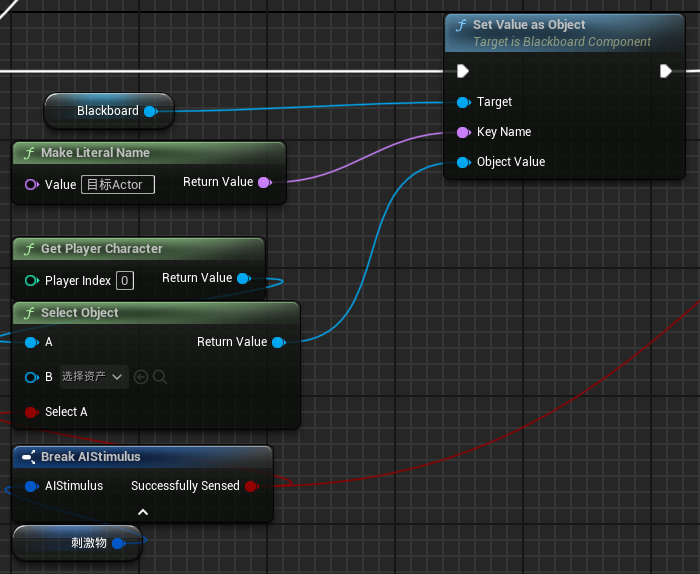
Stimulus是刺激源，

创建函数，命名为“处理视觉感知”，两个参数，和“目标感知更新时”的返回值一样。

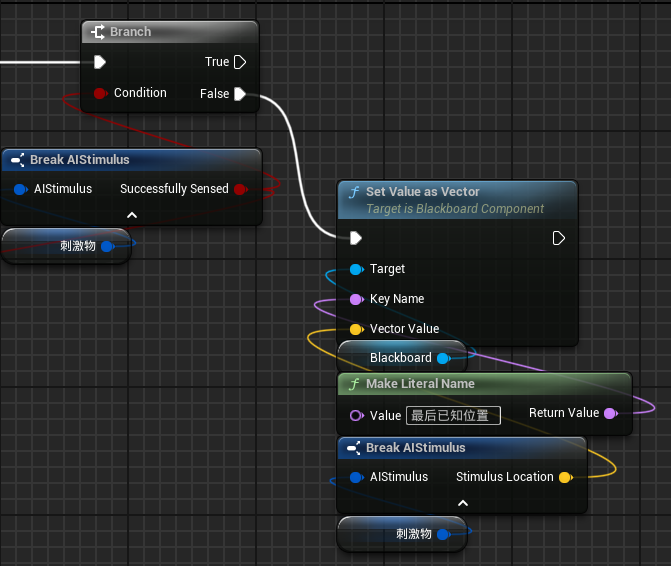


对Stimulus进行Get Sense Class for Stimulus从刺激物获得感官类，对返回值进行判断是否是视觉类。如果是AI感官视力，则看目标Actor是否是PlayerCharactor。如果是，则将目标Actor赋值给黑板中的“目标Actor”变量。





如果成功感知，则干他该干的，如果感知失败，则将刺激物的Location赋值给黑板的“最后已知位置”变量



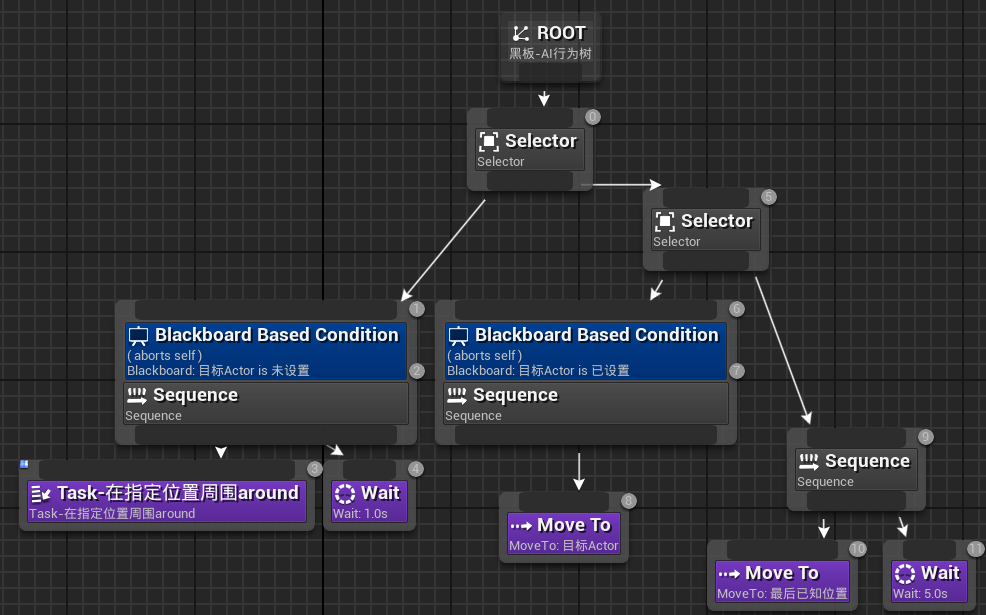
也就是说，黑板中至少会有三个变量：SelfActor（自带的）、目标Actor、最后已知位置。

目标Actor的键类型为Object，基类为Actor

最后已知位置的键类型为向量



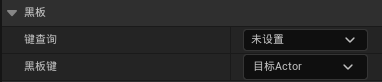
然后我们到AI行为树中。建立如下的行为树



先做说明，再做解释。

添加黑板节点只需要在指定节点右键选择添加装饰器在里面选择Blackboard即可。绑定的Blackboard可以在左键点击行为树中空白位置后在细节中赋值。

选择好Blackboard后，左键点击黑板，关注黑板的细节中“黑板”。



黑板中仅有两个选项：黑板键和键查询。用处是得知黑板键是否设置赋值。如果黑板键未被赋值，则走“未设置”的黑板的路径；如果黑板键被赋值，则走“已设置”的路径

现在要看流控制中的观察器终止。



我也不知道有啥用，选Self就对了，哈哈哈哈哈哈哈哈哈哈

根据上述描述，对行为树中的黑板进行设置

（行为树有自带的任务MoveTo，在它的细节中找到“黑板--黑板键”会发现里面有所有黑板变量。）



然后进入游戏进行验证。点击“’ ” 单引号”键可以查看AI行为树状态