行为树制作时的一些使用问题和解决方法

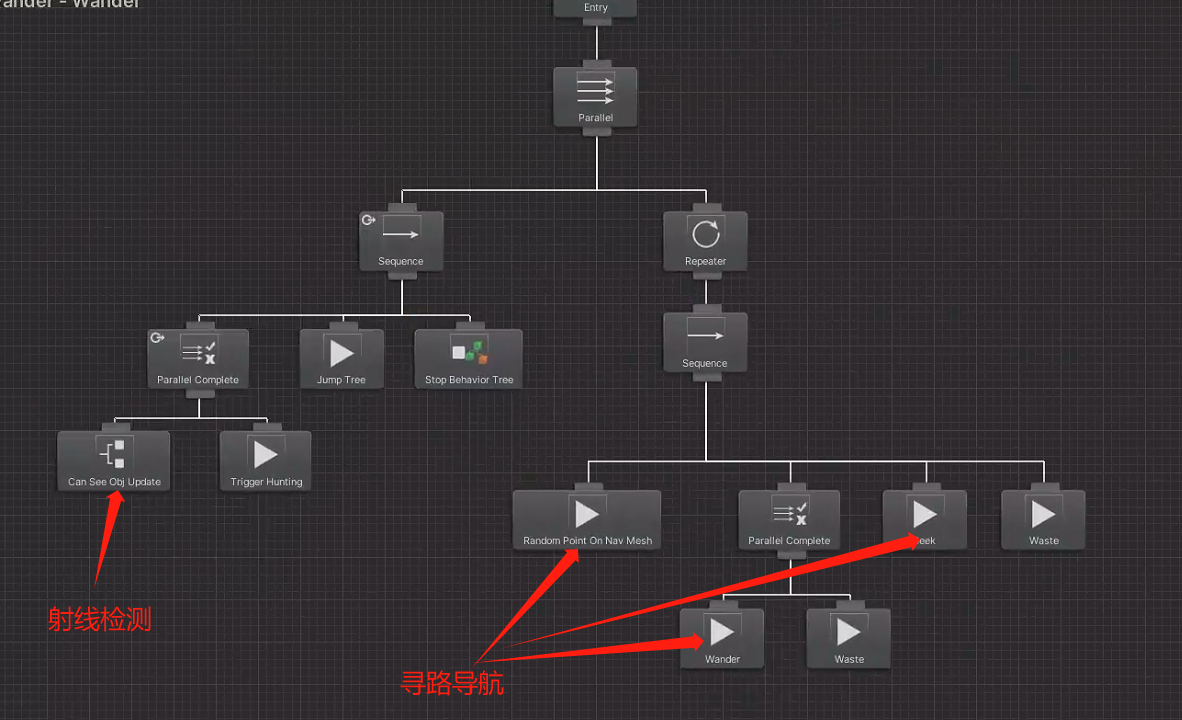
# BehaviorDesigner插件使用时遇到的问题及解决方案：

## 问题1

在使用该插件时，发现该插件的功能虽然好用，但却并不适合联网的游戏的使用，原因是该行为树的行为无法进行网络同步。

**解决方案**

大大缩小BehaviorDesigner插件的作用范围，在《驱鬼事务所》项目中，BehaviorDesigner插件目前只负责了做移动的导航和射线检测（即视线），如：



将其只在主客户端运行，由于移动只是transform的改变，所以只使用Photon同步位置即可，而检测到目标后将目标传到树外在进行后续操作。关于网络同步问题的详细操作，我会另写一篇文档详细介绍网络同步的操作。

## 问题2

插件中的NavMeshAgent并没有做出当位置在NavMesh外的判断，当位置点没有路径时会使导航做出奇怪的行为。该问题给出两种解决方法，视情况选择处理。

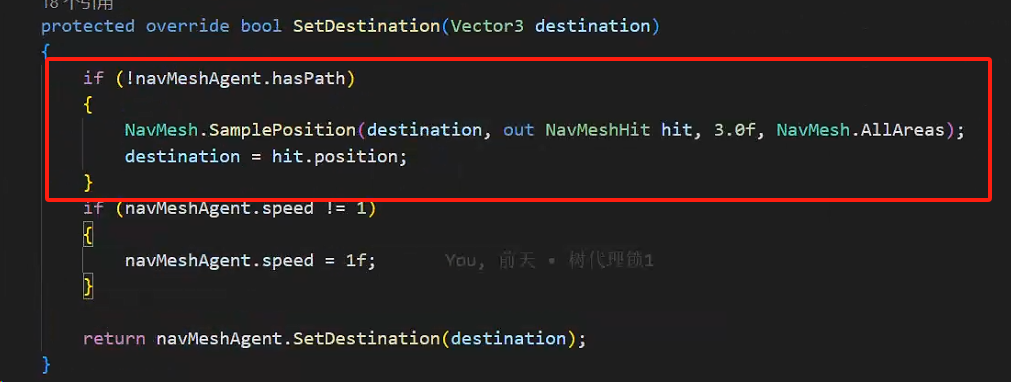
**解决方法**

**1.通过事件**

在NavMesh断开或者玩家离开NavMesh时给予通知，让AI根据事件通知做出其它操作，创建事件，根据情况执行事件，如在玩家离开时NavMesh时执行离开事件，并在玩家回归NavMesh时执行离开事件。具体操作例如，在目标（比如玩家）脚本定义一个Event，目标（比如玩家）在脚本中编写在进入某些范围时执行，还需要一个Event，负责在目标（比如玩家）进入AI可活动范围时执行，负责AI行为的脚本去订阅这两个Event，执行不同的行为。

**2.通过判断目标点**

NavMeshAgent默认会重新寻路，但如果位置过于偏僻，依旧会出错，所以需要在设置目标点之前手动判断一下目标点是否可用，且NavMesh尽量只有连续的一块。如：在将目标点给NavMesh前做出判断，判断目标点是否可用 。(注：该示例只是阐述通过判断目标点是否可用解决该问题，并不是一定需要按示例操作。)



作用是在没有路径时寻找附近最近的NavMesh。

[HasPath](https://docs.unity.cn/cn/2019.4/ScriptReference/AI.NavMeshAgent-hasPath.html): 如果代理有经计算可以到达期望目标的路径，此属性将为 true，否则为 false。

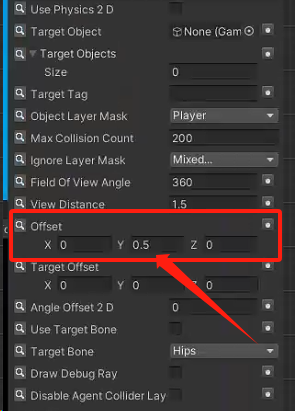
[NavMeshHit](https://docs.unity.cn/cn/2019.4/ScriptReference/AI.NavMeshHit.html): 导航网格查询的结果信息。

## 问题3

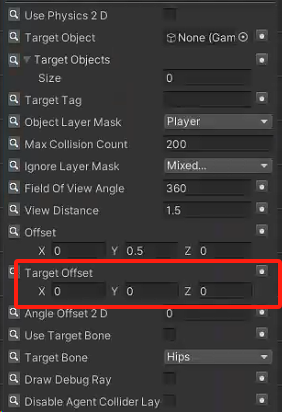
射线检测偶尔会检测不到目标，导致出现问题。

**解决方法**

BehaviorDesigner的射线检测是从原点开始的，一把人物模型的原点在最下方，所以会导致容易射线被地面挡住，可以使用属性面板中的offset将射线抬高



还有一种可能是检测到了目标碰撞体，但目标的原点在地下，这时也会被判定为检测不到，这时就需要抬高检测目标的偏移，即TargetOffset。(该处阐述的是在树上的解决方案)



## 问题4

BehaviorDesigner的一个bug是，一旦在树中使用代码终止这棵树，便会产生报错，虽然不会对程序造成影响，但终究是报错。

**解决方法**

这个并没有找到很好的解决方法，查阅文档及论坛发现都会遇到这个问题，并没有解决方案，所以我将其定义为一个bug。我使用的是插件中提供的Stop Behavior Tree节点终止这棵树

图形用户界面, 应用程序

描述已自动生成

若不在树中调用终止代码，而在树之外调用，不会产生这个问题。

# 只使用NavMeshAgent的导航，而不使用移动

## 使用原因

[由于NavMeshAgent的移动的碰撞采用的算法与物理碰撞不同，简单来说NavMeshAgent只会考虑避障而不会考虑碰撞](https://docs.unity.cn/cn/2020.3/Manual/nav-InnerWorkings.html)。NavMesh可以通过便是可以通过。而有时候我们会需要碰撞，这样随时更改添加避障十分消耗CPU性能，所以只是用NavMesh的导航，而不是用它的移动，而官方也有说明可以使用其它方法驱动，并提供了[相关API](https://docs.unity.cn/cn/2020.3/ScriptReference/AI.NavMeshAgent-velocity.html)及[部分注意事项](https://docs.unity.cn/cn/2020.3/Manual/nav-MixingComponents.html)。

图片包含 文本

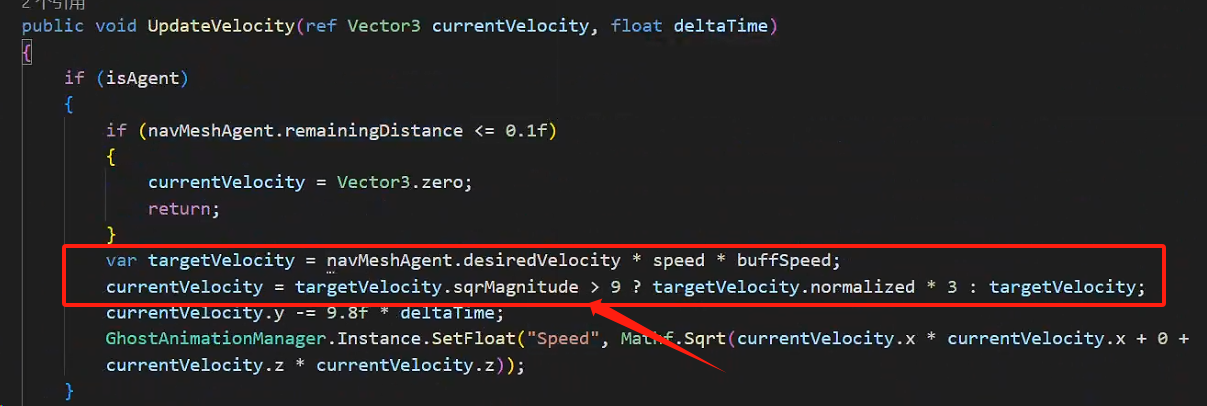
描述已自动生成

图形用户界面, 文本, 应用程序, 电子邮件

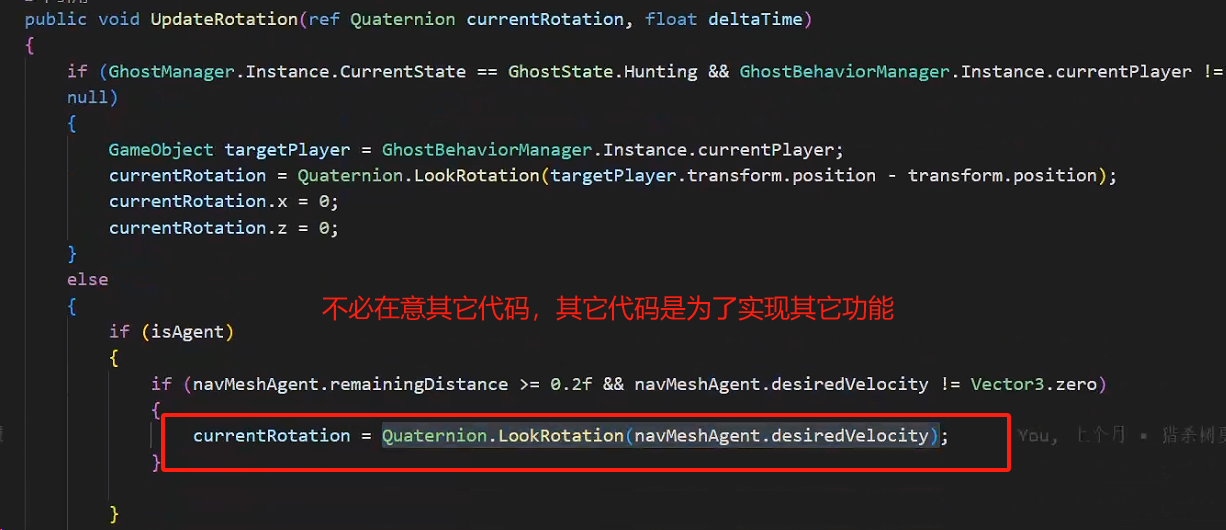
描述已自动生成

## 方法

将navMeshAgent.velocity = Vector3.zero，禁用掉NavMeshAgent的移动，通过[NavMeshAgent.desiredVelocity](https://docs.unity.cn/cn/2019.4/ScriptReference/AI.NavMeshAgent-desiredVelocity.html)来获取NavMeshAgent期望的速度向量，将其作为输入，由其它控制器控制移动，如使用KCC：



图中的三则表达式等判断，只是为了做一个速度上限，正常使用的情况下直接currentVelocity =navMeshAgent.desiredVelocity即可。这样还会出现一个旋转问题，也就是不旋转只移动，所以还需要根据navMeshAgent.desiredVelocity做出合适的旋转才行，直接使旋转等于Quaternion.LookRotation(navMeshAgent.desiredVelocity)，依旧以KCC为例：



其它代码是为了实现其它功能，只根据导航进行旋转使用Quaternion.LookRotation(navMeshAgent.desiredVelocity)计算旋转即可。

# 便于更换模型的技巧

有些时候第一次使用的模型可能不是最终模型，如果直接将脚本组件之类的挂在模型上，在更换是会非常麻烦，不亚于重新构建，所以建议将脚本组件等挂在一个空对象上，而模型作为空对象的子级，将模型孤立出来，这样不论是旋转还是移动缩放等，控制空对象子物体的模型的同样可以收到影响，在需要更换时直接更换子级的模型，而挂载着脚本和组件的空对象不需要进行更换，这样会方便很多。

