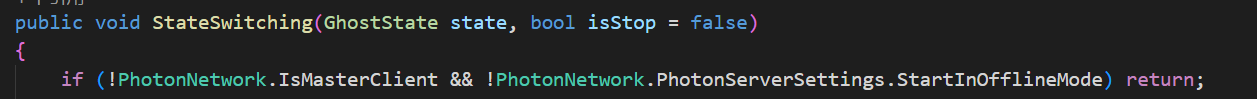
Behavior Designer网络同步和各客户端NPC数据的统一

# 1. Behavior Designer网络同步

**想法**：Behavior Designer无法进行同步，Behavior Designer插件中主要做的就是位置和动画等的改变，所以解决方法是只在主客户端进行运行，其余客户端只同步Transform和Animator，同时Behavior Designer中还包含一些全局变量等数据，这些的同步方法是单独提取出来存储，在脚本中单独进行同步。

## 1.1树的同步

在每次树启动时需要进行判断，只有主客户端才能允许启动。



在NPC上挂上Photon View相关组件，以实现行为和位置的同步。

图形用户界面, 应用程序

描述已自动生成

电脑萤幕的截图

描述已自动生成

Photon Animator View需要同步的还有Parameters的状态，负责容易导致非主客户端的动画状态机无法切换或者动画混合树无法正常播放等。

## 1.2树中全局变量的同步

Behavior Designer中存在全局变量，这在考虑主客户端退出时会导致全局变量的值丢失，所以也需要进行同步，由于Behavior Designer并不能同步，所以解决方法是新建脚本将这些变量在脚本中再存储一份。

屏幕上有字

描述已自动生成

使用Photon的IPunObservable接口来每个客户端同步这些数据，由于Transform和GameObject这类类型无法进行同步，所以这里我采用的是传递玩家ID的方式来解决。

文本

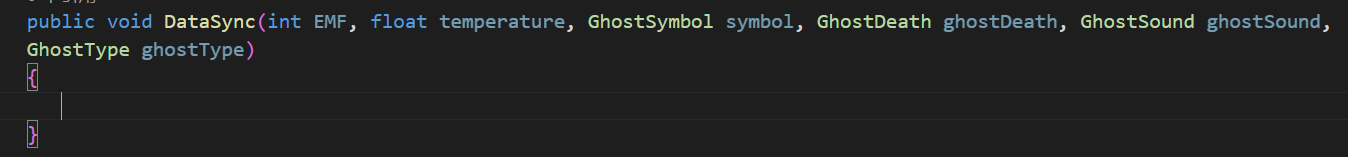
描述已自动生成

# 2.各客户端NPC数据的统一

思路：NPC数据我将其分为了两种，一种游戏开始定义后就不会改变的，一种是实时更新及不断改变的数据，两种使用不同的方式进行传递，不会改变的在游戏开始时数据初始化完成后发送一次RPC，将数据与各个客户端同步；而实时改变的数据则需要使用Photon的IPunObservable接口来进行实时的同步。

## 2.1开局发送RPC

创建一个用于同步这些数据的方法，参数即是所有需要传递的变量（Transform、GameObject等这类各客户端无法统一的变量无法进行传递）。



为其添加属性[PunRPC]，并在方法中将数据进行同步

文本

描述已自动生成

注：这里的data是我用来存储数据的一个数据类，并不是必须为数据类。

在数据初始化完成后由主客户端（数据在主客户端初始化）发送RPC并传入参数/

文本

描述已自动生成

这样就可以将所有客户端的数据统一化。

## 2.2实时同步

实时同步需要调用让类实现Photon的IPunObservable接口，该接口中有一个OnPhotonSerializeView的方法，负责数据的同步。

文本

描述已自动生成

直接在方法使用stream.Serialize(ref [要同步的参数])将数据进行同步即可。

文本

描述已自动生成

在这里需要实时传递的是当前的状态，由于我传入的是枚举，而枚举无法传递，我的解决方法是将枚举转换为int进行传递，同步后在将int转换为枚举进行存储。

# 主客户端切换问题

由于运行在主客户端，在主客户端退出时需要新的主客户端进行衔接

//TODO

# 瞬移问题

瞬移时其余客户端的同步是通过插值实现的，所以并不是真正意义上的瞬移，会出现位置错乱的问题。

//TODO