浮空轮

参考视频：

https://www.bilibili.com/video/BV1EeejesEvp?spm\_id\_from=333.788.videopod.sections&vd\_source=79fbe818ff96aae3677e123f0374bd91

https://www.youtube.com/watch?v=fIXmqlFfqLY

创建新的第三人称项目

创建GameAsset文件夹

创建FloatingWheel文件夹

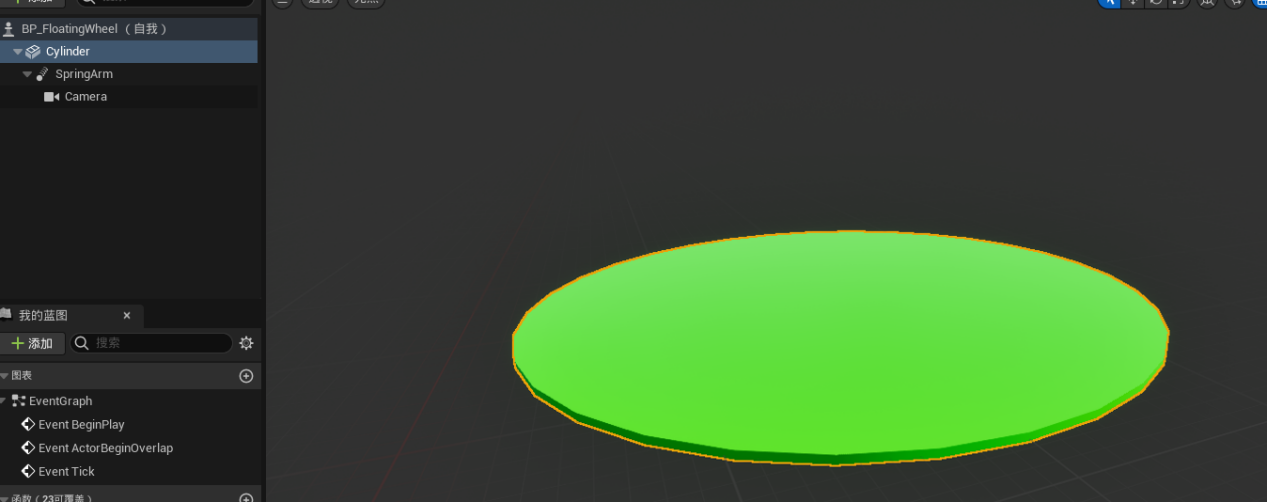
在文件夹中创建浮空轮的内容，创建Pawn类蓝图，命名为BP\_FloatingWheel

创建浮空轮的静态网格体。、

因为本人目前建模水平一般，用blender建模建了一个圆柱体fbx，导入到ue中



将BP\_FloatingWheel的静态网格体选择该网格体，给一个材质。



一定要从外部导入一个静态网格体，因为要设置角色在这个Vehicle上的位置，会用到插槽，后面会说到。

然后给他添加SpringArm和Camera，在SpringArm的细节中找到“摄像机设置--使用Pawn控制旋转”勾选

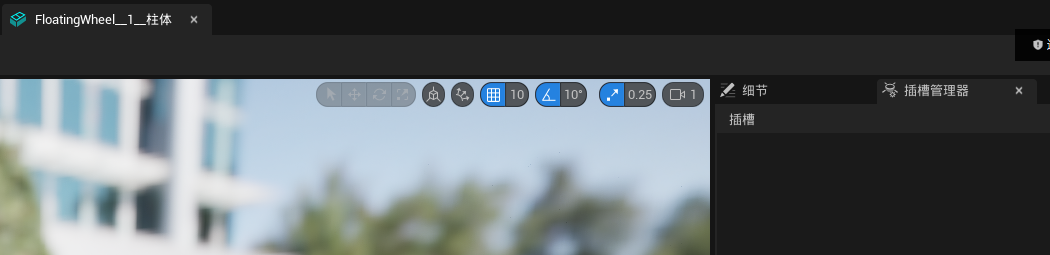


这样就简单实现了“车辆”。接下来就要实现角色如何和车辆进行通信。为此，我们会使用一个接口。

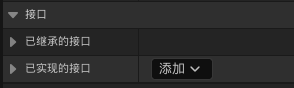
在FloatingWheel文件夹中创建“蓝图--蓝图接口”，命名为“BPI\_Vehicle”。基本上每个车辆都会有这个接口。打开蓝图，创建第一个函数，实现进入车辆功能，命名为“EnterVehicle”。

在实现浮空轮上的EnterVehicle函数之前，我们需要给角色创建一个插槽（Socket）。这个插槽基本上就是角色在浮空轮上的位置。

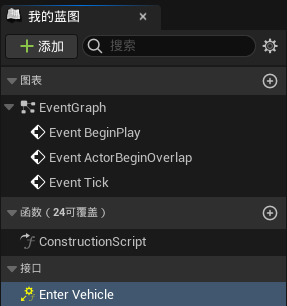
打开静态网格体，在右侧有细节和插槽管理器。在插槽管理器中添加一个插槽，命名为CharacterLocation。适当设置他的位置，这是角色在浮空轮上的位置。



然后我们给BP\_FloatingWheel添加接口。在BP的类设置中，找到接口，添加上我们刚刚实现的BPI\_Vehicle接口。



然后在“我的蓝图”中双击刚刚添加的接口



就会在事件图表中出现如下的节点

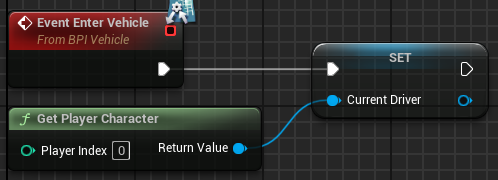


接下来的内容实现，要在BP中的EnterVehicle中实现，而在蓝图接口中的该函数先不管它。

首先尝试在BP\_FloatingVehicle中获取驾驶人，也就是BP\_Character

在BP\_FloatingWheel中创建变量，命名为CurrentDriver，类型为Character（不是第三人称自带的那个）。

接下来，我们需要在EnterVehicle的时候SetCurrentDriver

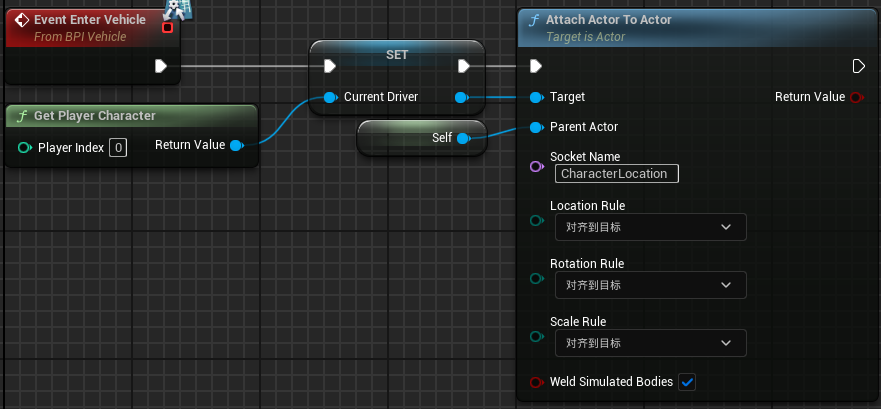


然后将这个对象引用应用在FloatingWheel上，应用到插槽上，用到**Attach Actor To Actor**节点

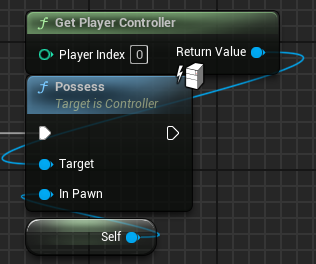


Target就是Driver，ParentActor就是浮空轮Self，SocketName就是在静态网格体中添加的插槽

其他的选择如图所示



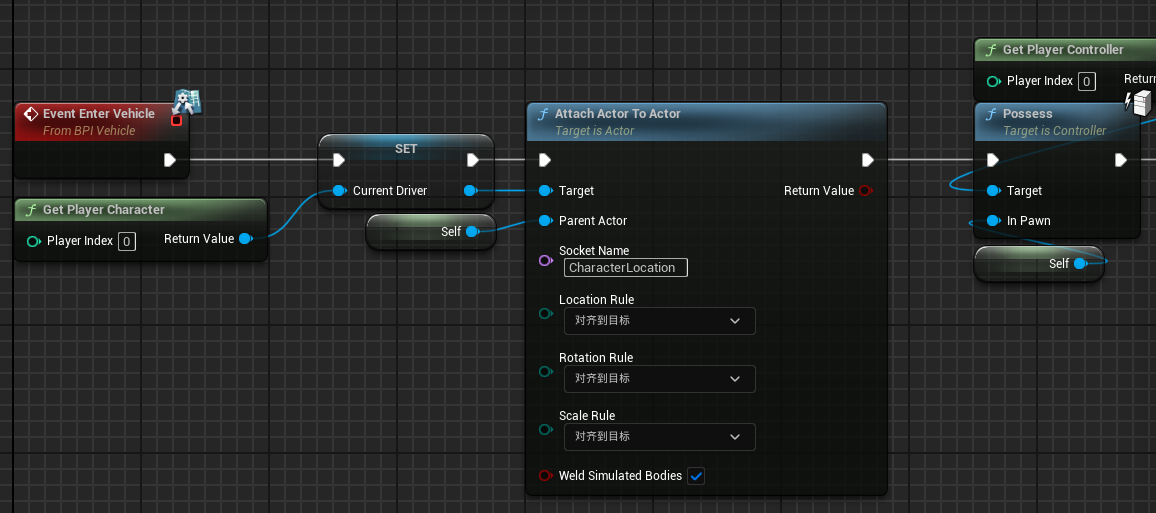
这样角色就附着在浮空轮上了。接下来实现浮空轮的控制。为了实现控制，我们需要用到玩家控制器来控制。而BP\_FloatingWheel的父类是Pawn，可以控制。也就是GetPlayerController的Possess函数

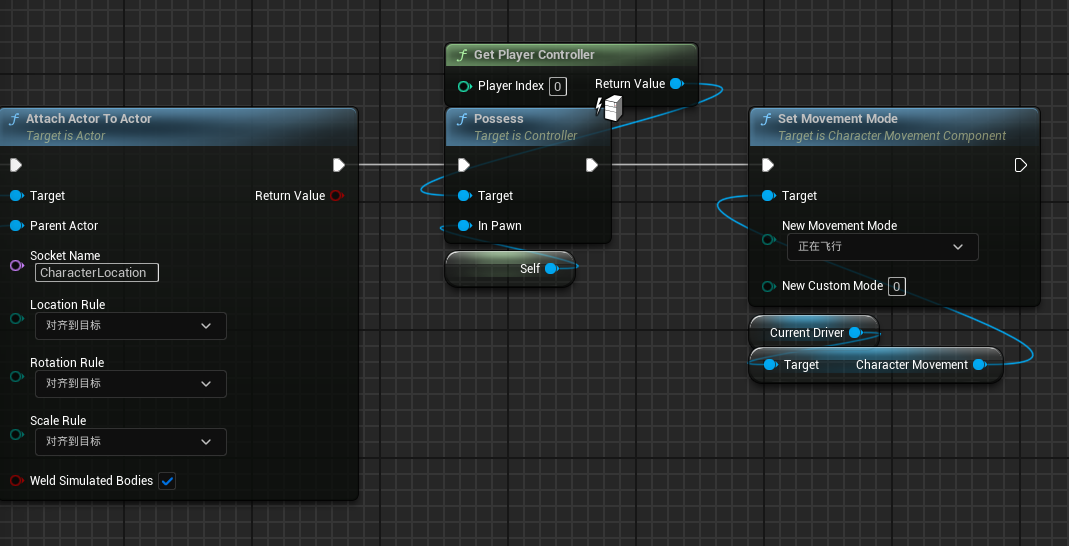


然后设置Driver的移动方式。在Character类的CharacterMovement中提供了多种移动方式



这里选择正在飞行。这样角色蓝图得到移动组件就不会不断尝试将角色放到地面。





接下来实现下浮空轮

首先在BP\_FloatingWhell的视口中添加一个箭头作为位置参考，以便我们可以退出。

在组件中添加一个箭头，命名为ExitLocation，作为位置参考，来表示我们将角色放置的位置。

接下来我们就要完成控制器控制角色上下。创建输入操作，命名为IA\_Interact。默认的bool就可以。然后打开输入映射上下文，添加该输入操作。，选择E键。

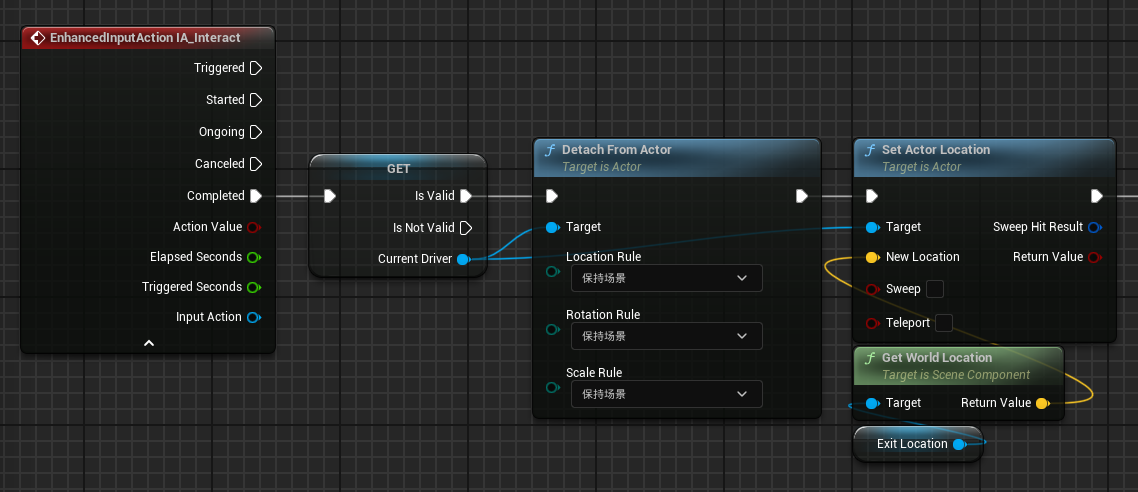
接下来回到BP\_FloatingWheel，实现退出功能。

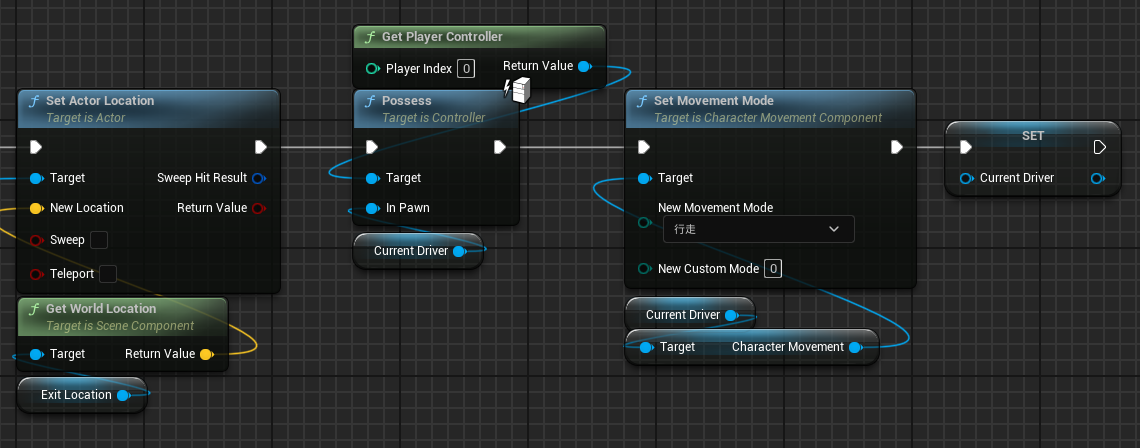
退出的核心函数就是Detach From Actor



这时就已经将Target（Driver）从浮空轮上分离。接下来只需要将Driver的位置设置为出口位置。也就是将ExitLocation组件的GetWorldLocation赋值给Driver的SetWorldLocation

然后控制PlayerController的Possess为Driver，并将角色的CharacterMovenent的移动模式改为行走，并在之后清除Driver变量。





现在角色有能力上下车了（在BP\_车中实现），现在只需要在角色蓝图中调用EnterVehicle函数（来自蓝图接口）

对于角色，首先需要一种与事物交互的方式。一般来说，需要集成一种交互系统。但是现在还没有交互系统的经验，所以独立创造一个交互系统。

下面将使用球体碰撞与世界交互。

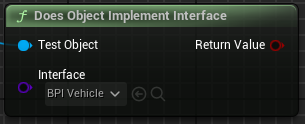
在角色蓝图的组件中添加球体碰撞，命名为InteractionArea，适当调整大小（这里选择200）。

我们下面就简单的使用这个球踢作为交互系统。

接下来创建OnComponentBeginOverlap事件，当OtherActor为BP\_FloatingWheel时，为了后面赋值方便，将该BP\_FloatingWheel创建变量。

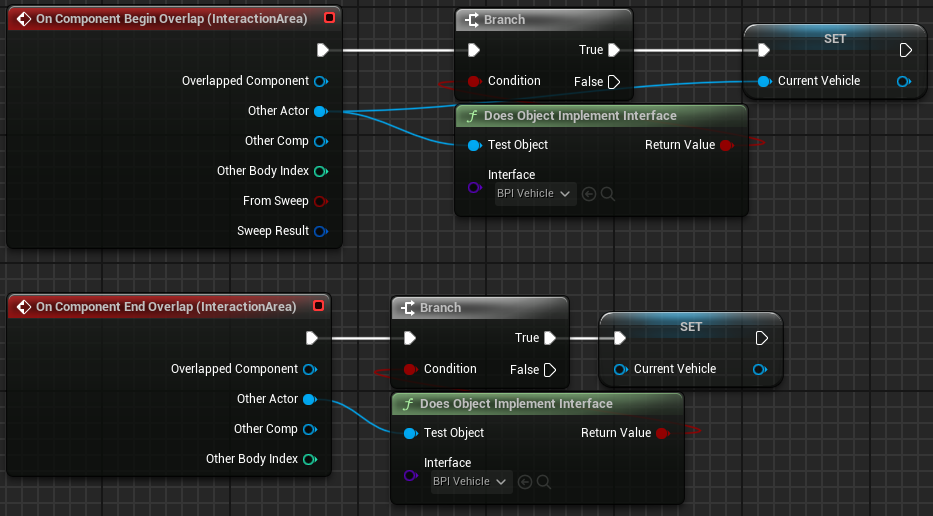
这里我们不用CastTo节点，我们用蓝图接口实现。

我们需要询问OtherVehicle是否实现了车辆接口，即Does Object Implement Interface节点



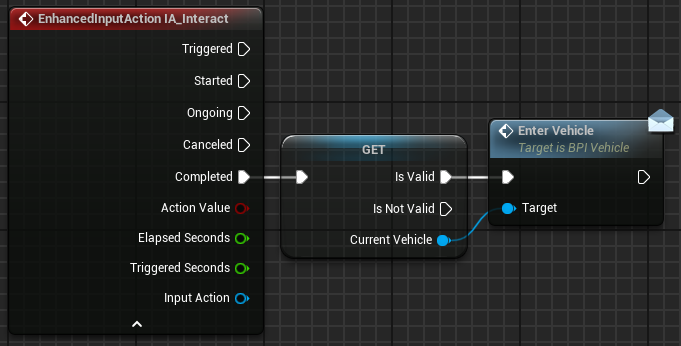
如果为真，则将OtherActor设置为变量，命名为CurrentVehicle。注意，CurrentVehicle变量是Actor类型。

接下来添加OnComponentEndOverlap事件节点。如果EndOverlap，我们就要清除CurrentVehicle



然后实现按键交互。

我们是否靠近车辆，我们可以使用当前车辆变量。如果这个变量是合理的，我们就会调用CurrentVehicle的EnterVehicle（事件）函数



现在我们就可以在游戏中摁E进入浮空轮和退出浮空轮。

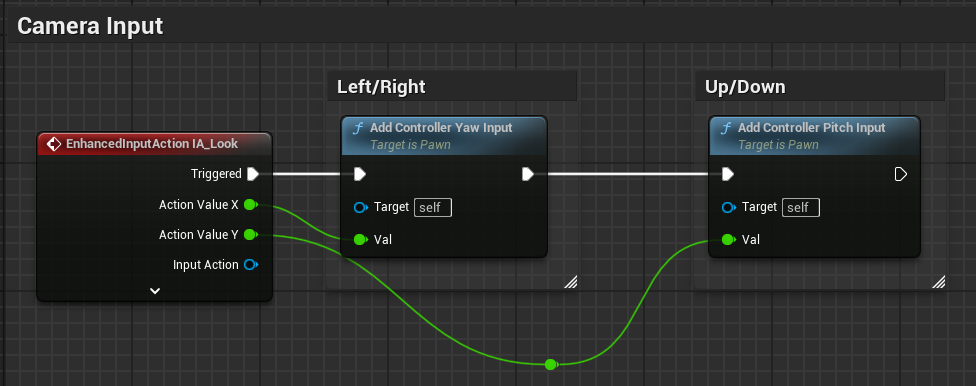
但是摄像机不对。

原因是车辆的SpringArm碰到了角色。

所以我们要在角色蓝图中自定义碰撞预设。将Camera忽略。需要对胶囊体组件和网格体组件都进行设置。



但是现在并不能控制浮空轮的运动。所以我们要实现他的运动。也很简单，将角色蓝图中的相关节点族复制到BP\_FloatingWheel中即可。



但是目前只能复制鼠标转动摄像机的功能。对于其他的功能，还需要手动实现。

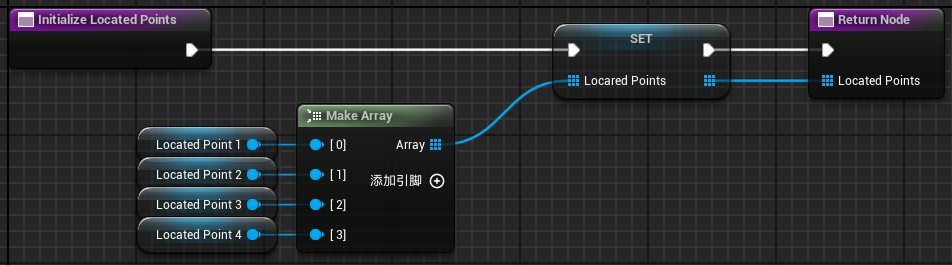
现在我们给浮空轮添加四个点作为位置参考，来正确的应用力。

创建4个箭头组件，方向向下。

接下来我们需要一个数组，这样可以遍历所有这些点并应用力。

在BP\_FloatingWheel中添加变量LocatedPoints，变量类型为SceneComponent的对象引用的数组。

然后创建函数，将四个点放到该数组中。



有Set，所以不能是纯函数。

接下来在EvnetBeginPlay时调用该函数。

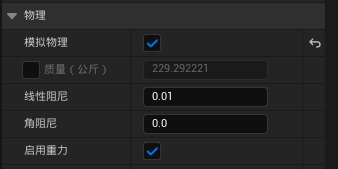
接下来开始计算悬停力。

添加变量，命名为HoverForce，变量类型为float型的单个。

添加变量，命名为HoverAcceleration，float型

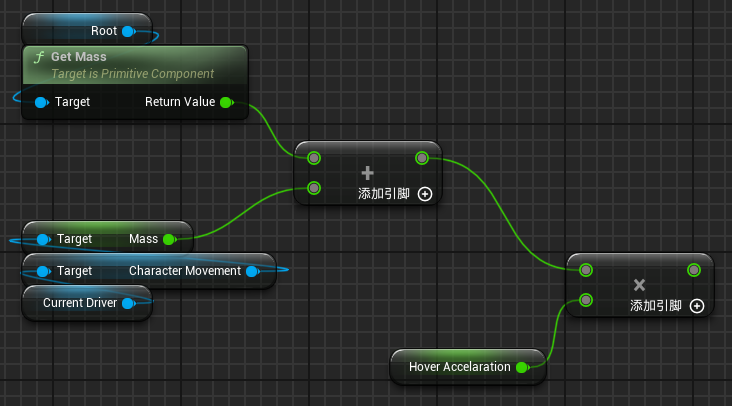
添加函数，命名为CalculateHoverForce。计算悬停力，需要F=ma。

找到根组件，在细节中找到“物理--质量（公斤）”。我们可以不勾选，让虚幻来自己默认。为了获取质量，我们需要勾选“模拟物理”

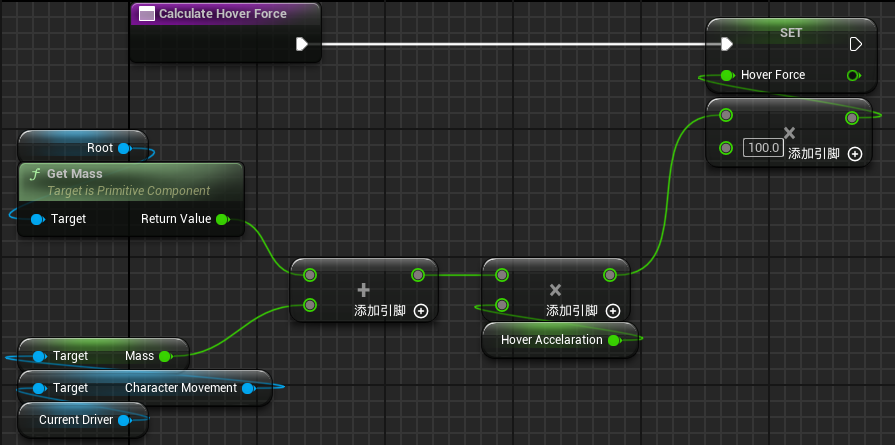


为了获得质量，将静态网格体组件（应该就是根组件）拖入到事件图表中，通过Get Mass函数来获取质量。

但是现在我们还需要Driver的质量。Driver的质量需要从CharacterMovement组件中获取。

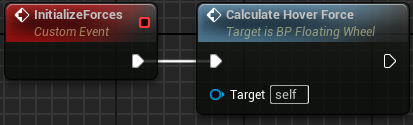


然后就是量纲的问题。这样既算出来的力的单位是N，虚幻里力的单位是特有的。转换起来很简单，将计算结果\*100就是。

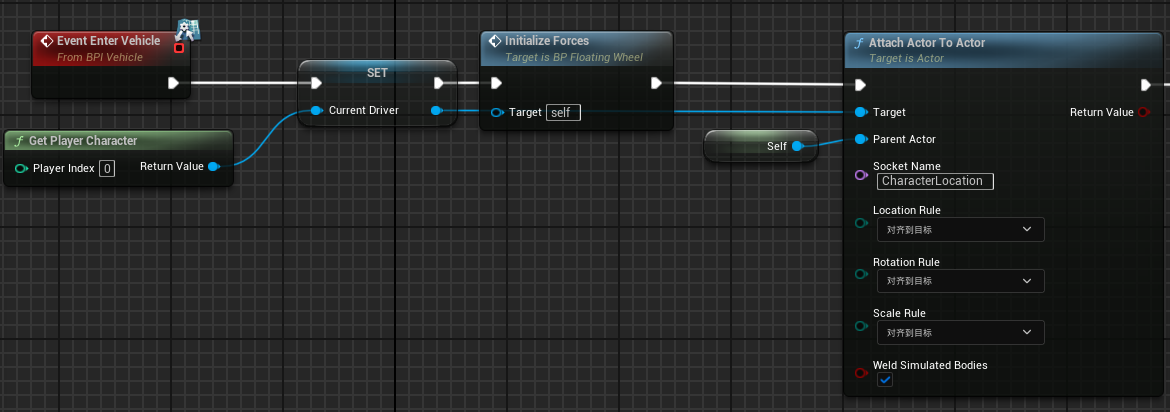


现在我们需要给加速度一个默认值。这里给200。

为了给浮空轮进行力的计算，我们自定义事件，命名为InitializeForces。



并在EnterVehicle时调用此事件。



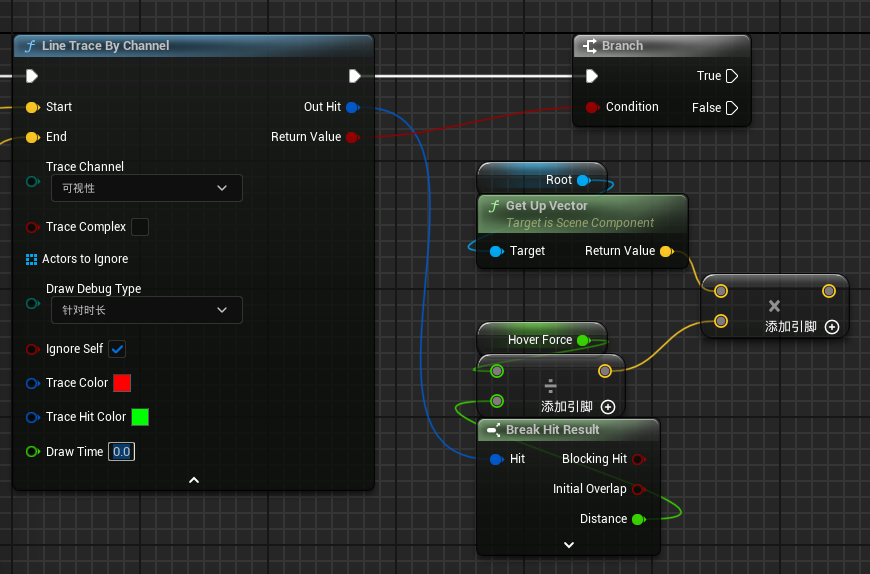
接下来在悬停点上使用Force。

创建变量，命名为HoverDistance，表示悬停距离。创建变量，命名为TurnOn，表示浮空轮打开并应用力。

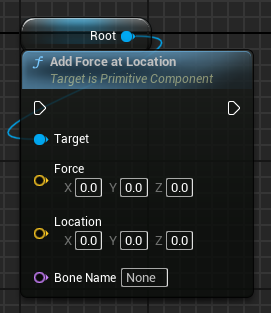
接下来创建一个函数来应用HoverForce。创建函数，命名为ApplyHoverForce。

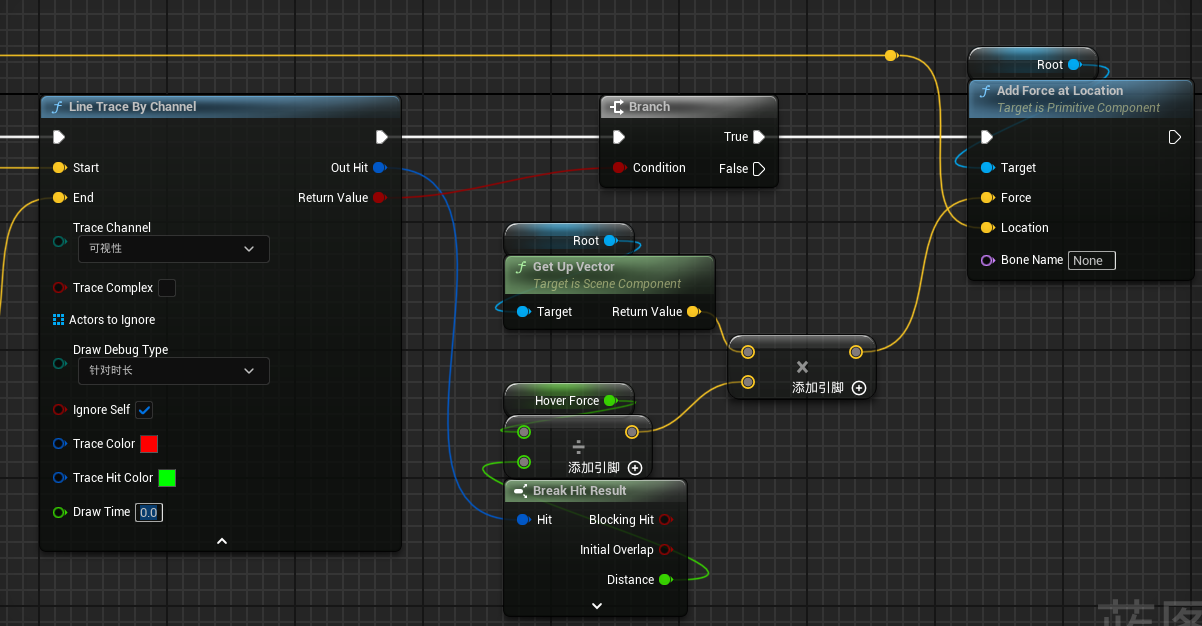
首先要看浮空轮是否打开，打开则遍历定位点。给每个定位点LineTraceByChannel来看我们是否接近地面。Start参数即为定位点的GetWorldLocation。End参数即为“向下的单位向量\*HoverDistance+定位点WorldLocation”的结果。

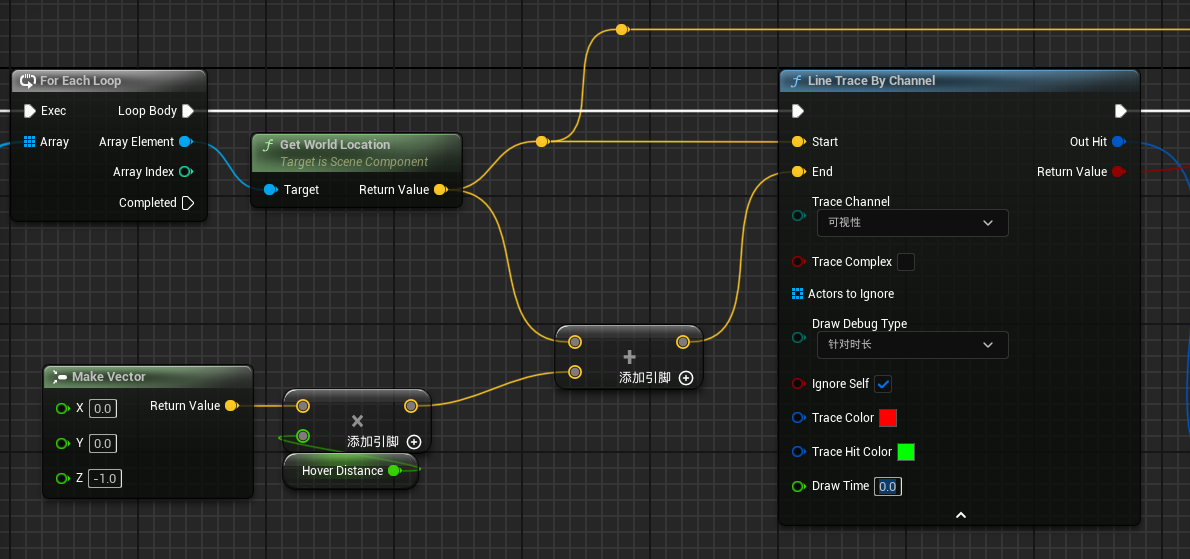
如果检测结果，过于接近地面，我们希望可以增加力，



让我们要应用力的时候，







然后回到事件图表，在EventTick时调用该函数，并在EnterVehicle和Exit时设置TurnOn变量的值。

同时也要设置HoverDistance的值。

现在就可以进入游戏，会发现浮空轮非常不稳定。下面就来让他变得稳定。

要稳定悬浮板，首先要创建一些新的变量。

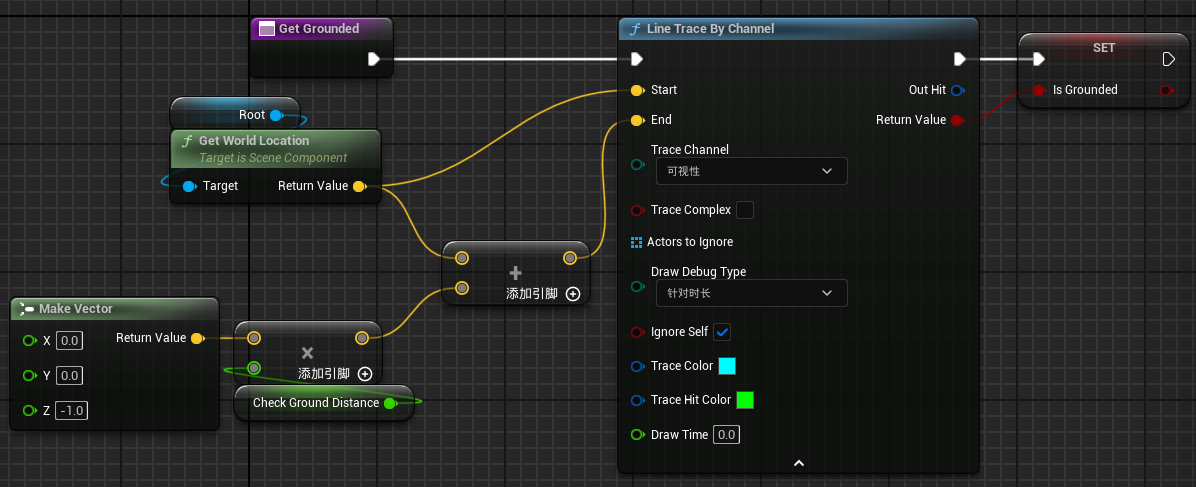
创建变量，命名为LinearDampingForce，float类型，表示线性阻尼力。默认100

创建变量，命名为AngularDampingForce，float类型，表示角度阻尼力。默认5

创建变量，命名为MaxRotationAngle和MinRotationAngle，float类型，默认±45

接下来应用线性阻尼。在应用线性阻尼之前，需要知道我们是否在地面上。创建函数来实现，因为想要一个不同的距离来检查我们是否在地面上或者不在

创建函数，命名为GetGrounded。创建两个变量，命名为IsGrounded和CheckGroundDistance，默认值为250。



这个函数用于告诉我们，我们是否在地面上

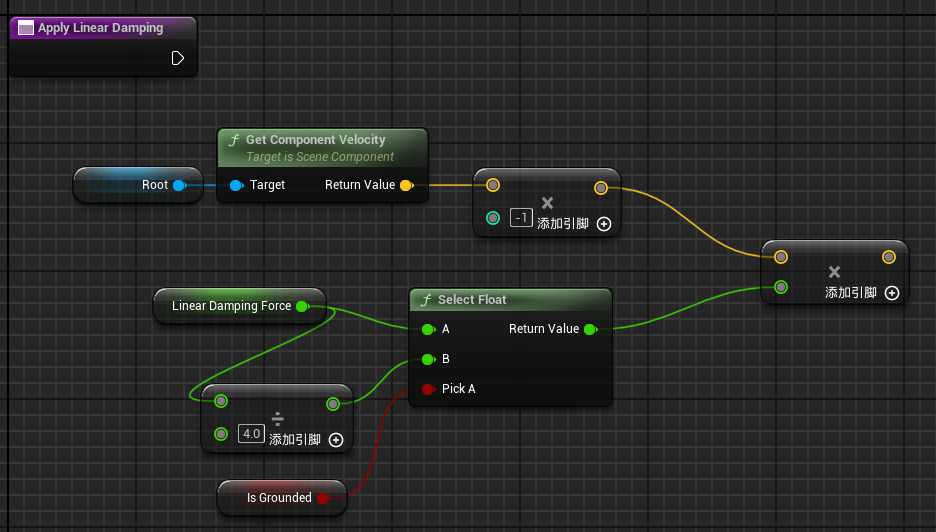
然后在EventTick中先调用该函数



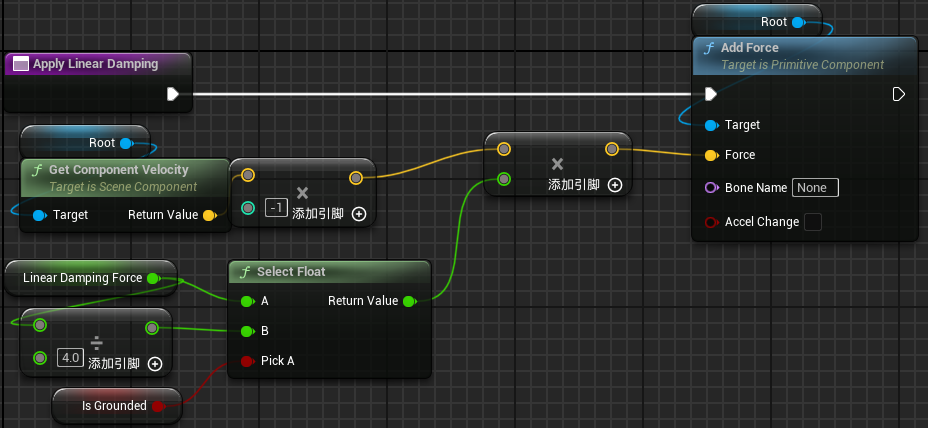
然后创建函数来应用线性阻尼，命名为ApplyLinearDamping

首先获取移动方向。我们需要应用与速度方向相反的线性阻尼。然后就需要应用我们创建的线性阻尼力，在速度反向上乘以线性阻尼力。

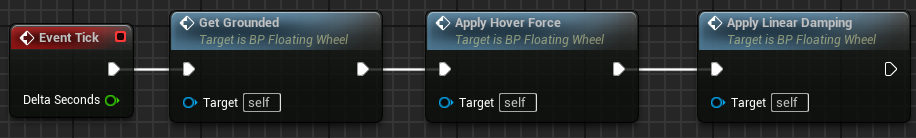
我们希望线性阻尼随是否在地面上而改变，所以Select float。当不在地面上的时候，我们希望力会更小。



然后将计算结果对Root进行AddForce

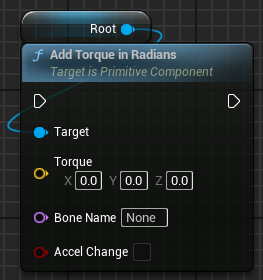


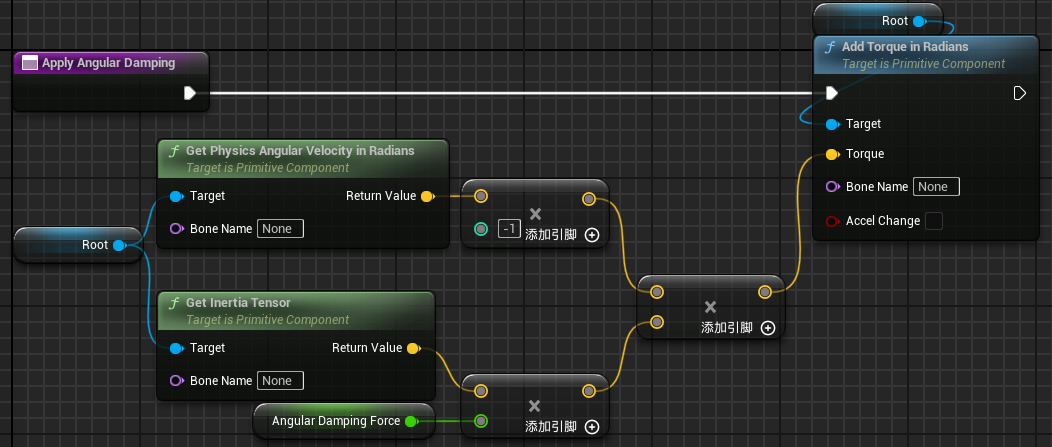
然后在EventTick中调用该函数



同时也需要应用角阻尼。

创建函数，命名为ApplyAngularDamping。原理是一样的，只是需要获取的速度是Get Physics Angular Velocity。还有一点就是要将角阻尼力乘以悬停板的惯性张量，即Get Inertia Tensor。这样就有了应用于悬停板的扭矩。也就是Add Torque节点

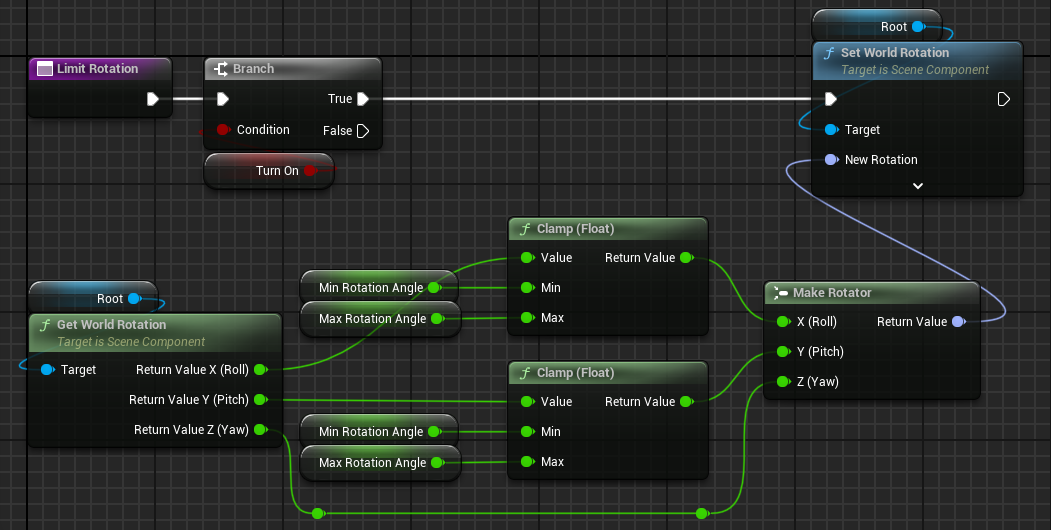




然后在EventTick中调用该函数

现在我们需要限制浮空轮的旋转比如限制旋转的俯仰和翻滚（Pitch和Roll），当然也可以不限制，这里给出如果限制该怎么做。

创建函数，命名为LimitRotation



在EvenrTick中在应用角阻尼之后调用LimitRotation函数

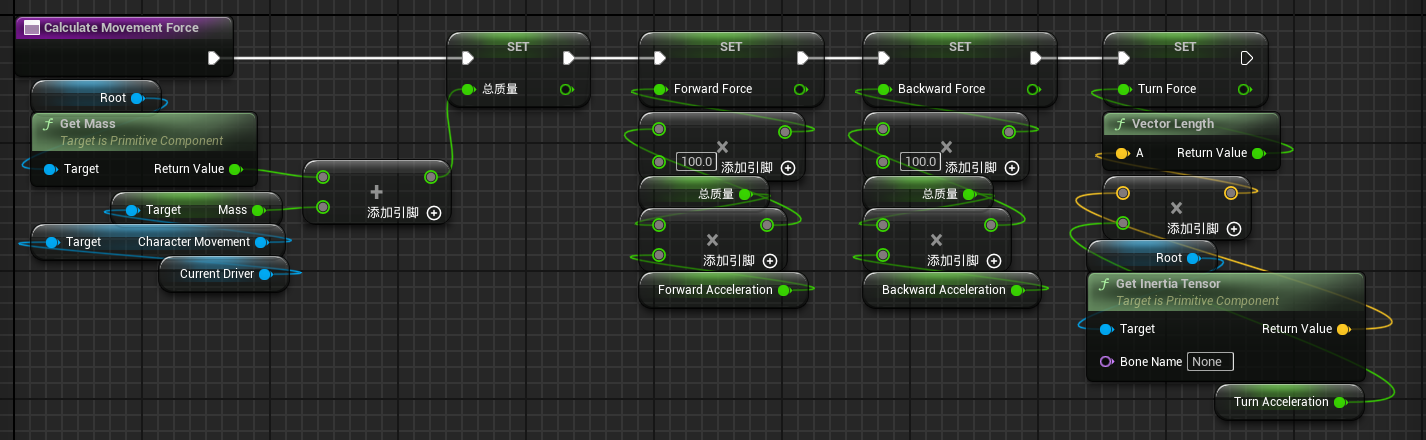
这时就可以比较稳定的使用浮空轮了

接下来实现浮空轮的控制移动。

要做运动，首先创建几个变量。

创建变量Forward Acceleration、Backward Acceleration、Forward Force、Backward Force、Turn Acceleration、Turn Force。都是float型

创建函数，命名为CalculateMovementForce。下面会用到牛顿第二定律来计算。

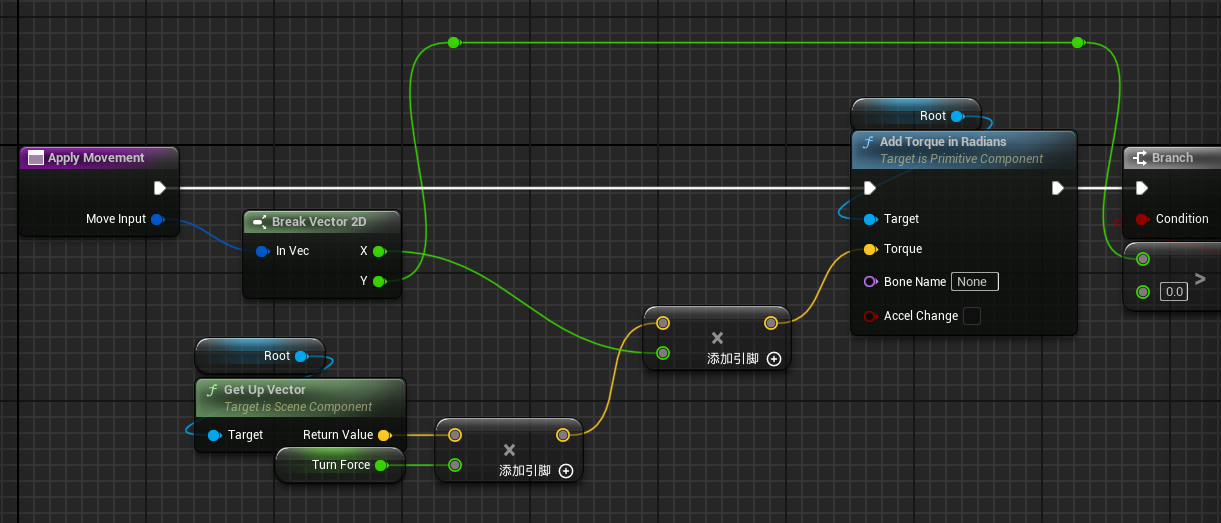


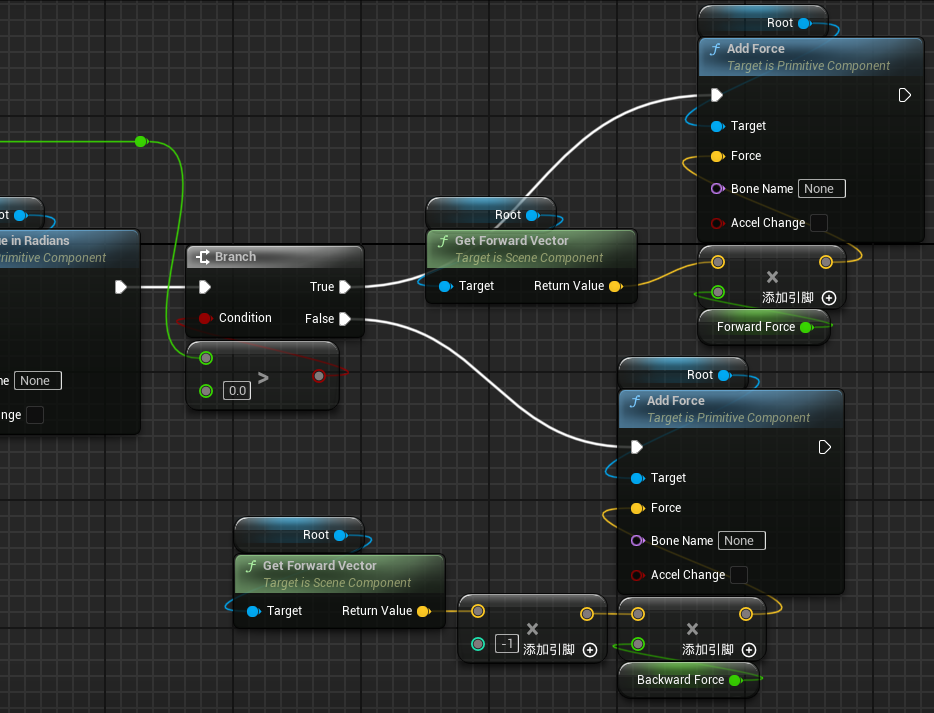
下面将会调用该函数。我们会在InitializeForce时调用该函数，在CalculateHoverForce之后调用该函数。

接下来创建将运动应用于浮空轮的功能。

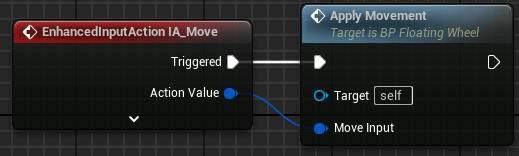
首先简单设计一下，AD两个按键用于转动浮空轮

创建函数，命名为ApplyMovement。添加Vector2D类型的形参，命名为MoveInput





然后调用增强输入的IA\_Move事件节点。



然后给三个加速度一个默认值，前为10，后为5，转为15。

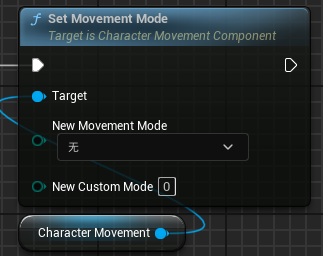
修一个bug，有时上车之后摄像机视角会出问题。

为了复刻该bug，可以将浮空轮倒置，然后就能触发该bug

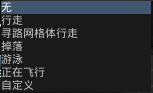
下面修好他。

在BP\_FloatingWheel中给SpringArm的“摄像机设置--继承Roll”关掉即可

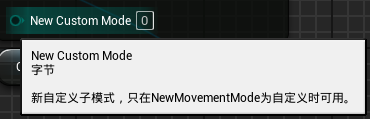
Set Movement Mode



虚幻引擎自带的移动方式模组。有多种移动方式。



首先是自定义。自定义与函数的最后一个参数相关。



通过BP\_ThirdPerson进行实验。