p1 FPP, viewUpDownRotation

1. true first person:

attach cameraBoom to skeletonMesh, set socket like neck\_02(to seamless view)

just set cameraBoom-length=0,then adjust camera to proper pos

enable **use pawn controller rotation** in camera（ **make sure that camera remains fixed to rotation of pawn**.）,

on bp\_tppCharacter, enable **use controller rotation Yaw**

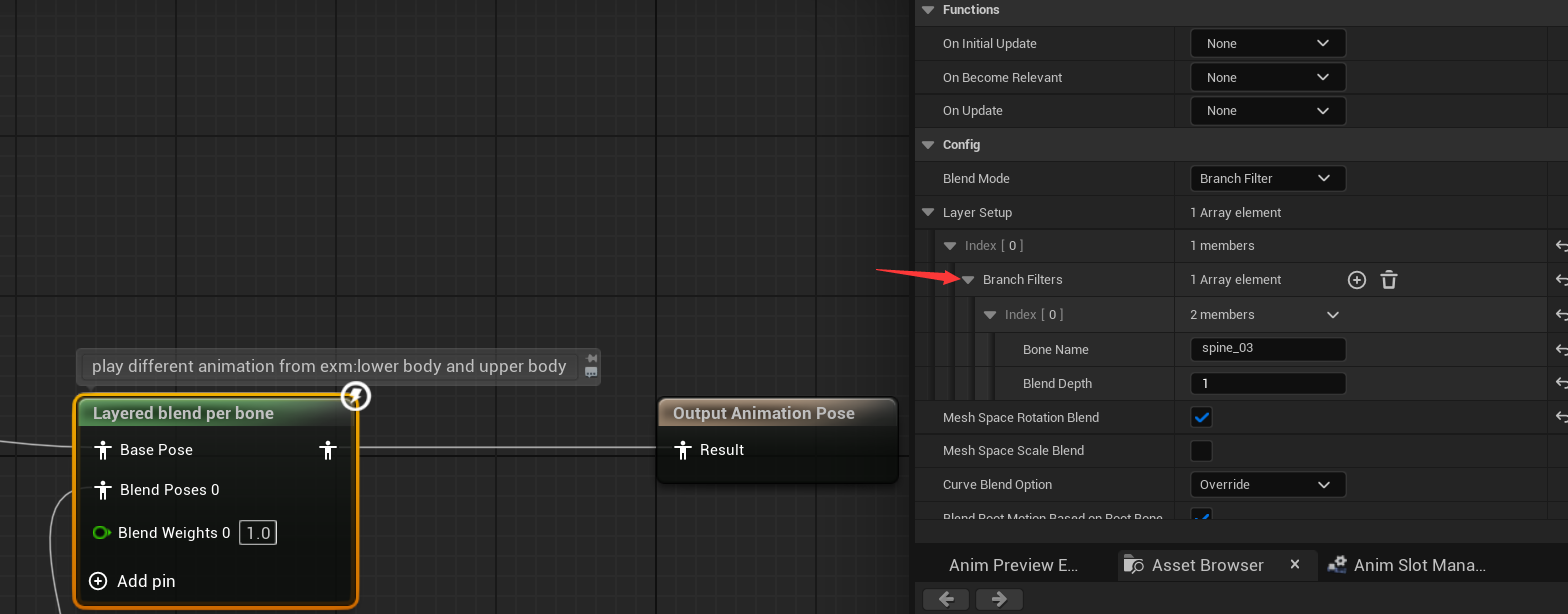
to full control over the direction

\*cameraBoom:摄像机杆，spring arm：弹簧臂

2. assets retargeting

3. layered blend per bone: play different animation from exm:lower body and upper body

3.1

3.2 specify Branch Filters:

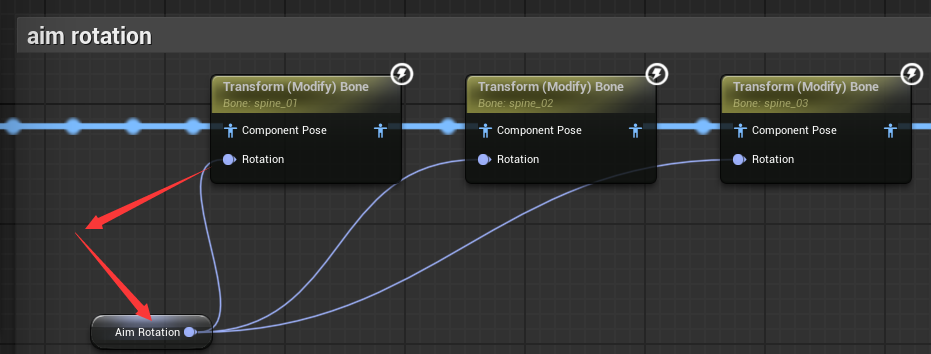
to clear where to split the upper and lower body

3.3 disable “orient rotation to movement” in character movements, to make sure no acc factors

4. to make body rotation(look up and down rotation):

4.1 make three “transform bone modify” nodes and specify Spine\_01 02 03

and we want “rotation” pin available, and make a rotation variable or promoted one like arrow



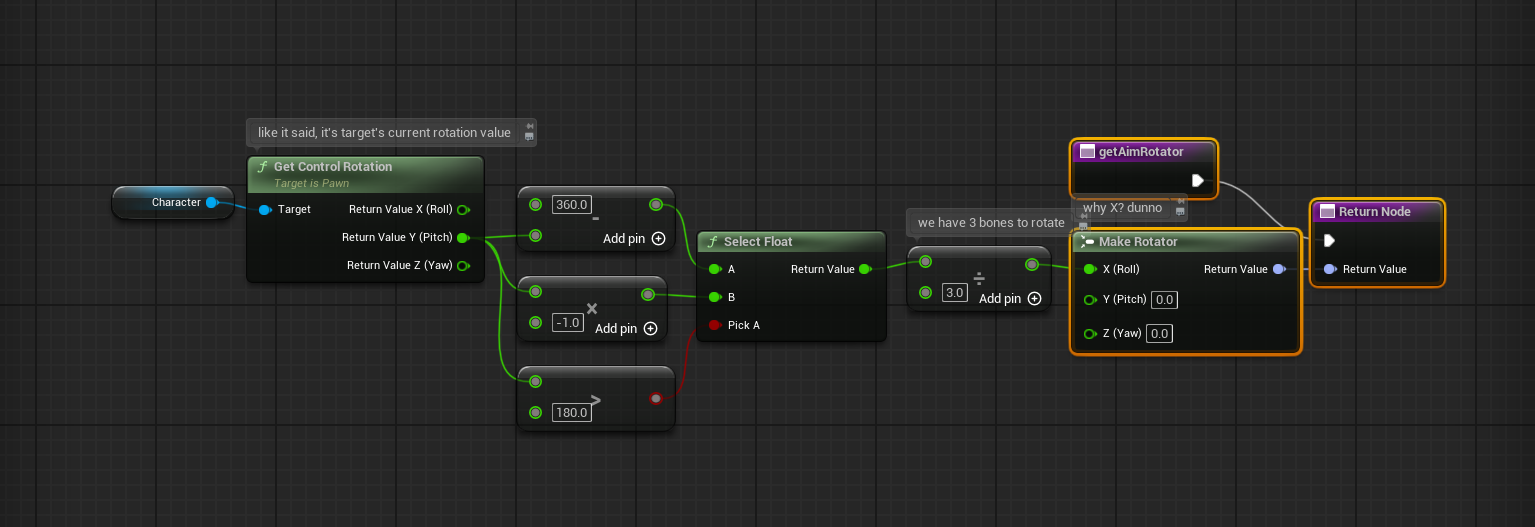
\*insert after “mainstate” node of course

\*inside the ABP\_manny’ animGraph

4.2

(pitch, yaw, roll=俯仰、偏航、滚转)

go to eventGraph and make a update upNdownRotation variable function like pic below, it’s quite understandable so



explanation:

### **为什么要做 360 - Pitch 或 Pitch \* -1？**

这是为了把 Pitch（摄像机抬头或低头的角度）转换成适合骨骼动画的“前后倾角”。

在 UE5 中：

* Pitch（上下视角）从 -90 到 +90 是比较常见的可控范围。
* 但 Control Rotation 的 Pitch 实际范围可能是 0 ~ 360（取决于你怎么看），所以：

| **摄像机动作** | **Pitch 值** | **目标 spine 弯曲方向** | **最终 rotator** |
| --- | --- | --- | --- |
| 抬头 | 小于 180（如 45） | 往后仰 | -15（负值） |
| 低头 | 大于 180（如 270） | 往前弯 | 90（正值） |

所以：

* **抬头时 → Pitch 小于 180 → 乘以 -1 得负数 → spine 向后弯**
* **低头时 → Pitch 大于 180 → 360 - Pitch 得正值 → spine 向前弯**

这样就把 0 ~ 360 的值，统一映射到了一个逻辑上正确的 “弯曲角度” 范围里。

再 **除以 3** 是为了减小动作强度，避免 spine 弯得太夸张。

### **❓为什么使用 Make Rotator 的 X（Roll）？**

Transform (Modify) Bone 中用的是 **Rotation**，而 Make Rotator 是用 XYZ 分别代表 Roll、Pitch、Yaw：

| **Axis** | **Rotator字段** | **意义（通常）** |
| --- | --- | --- |
| X | Roll | 左右翻滚（你这里当成上下弯曲） |
| Y | Pitch | 上下抬头低头 |
| Z | Yaw | 左右转头 |

虽然我们是根据“Pitch（摄像机上下）”来判断视角，但因为 spine 是绕 **X 轴（Roll）** 来旋转弯曲的，所以要把结果写入 Make Rotator 的 X。

所以这是个 **轴映射转换**的过程：  
 我们从 Pitch 推出 “想要弯曲多少”，但要通过 Roll 去实现。

### **✅ 总结**

你这段逻辑是合理且高效的，以下几点你可以记住：

* Control Rotation 的 Pitch 不是简单的 -90~90，可能跨过 180°。
* spine 往前或往后弯，要通过 Roll 控制（X轴）。
* 做一些变换（如 360 - Pitch）可以让动作方向更直觉。
* 除以 3 是为了更自然的力度，避免“突然弯太多”。

p2 addGun, leftHandAttach

1.1add some weapons

1.2 Create a blueprint class in “actor” form

1.3 open it and add、apply a skeleton mesh

2 attach to character in bp\_tpp, apply a socket (create one in character’s mesh)

3 adjust leftHand, make it natural by attach foreGrip

\*

| **空间类型（Transform Space）** | **定义一句话讲清楚** | **适用场景（什么时候用）** |
| --- | --- | --- |

| **World Space** | 世界坐标，参考的是整个世界的原点 (0,0,0) | 控制目标点在世界某个固定位置，比如角色走到哪都不变 |
| --- | --- | --- |

| **Component Space** | 相对于角色自身 Mesh 的原点 | 控制目标点在角色局部空间中，比如武器永远在角色胸前 |
| --- | --- | --- |

| **Parent Bone Space** | 相对于父骨骼的位置与旋转 | 一般用得少，多用于复杂层级骨骼控制 |
| --- | --- | --- |

| **Bone Space** ✅ | 相对于某个**指定骨骼**（Effector Target）的局部坐标 | 目标位置是某个骨骼下的 socket，比如左手去握右手上的握把 socket |
| --- | --- | --- |

**当你要把左手吸附到右手上的某个 Socket（如 leftHandSocket）时，传入的 Transform 是以 hand\_r 为参考点的，所以要用 Bone Space，并把 Effector Target 设置为 hand\_r！**

\*two bone ik and FABRIC

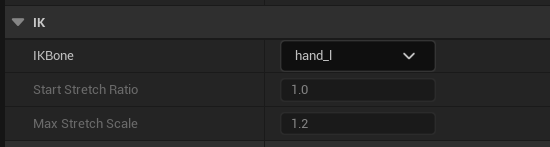
ue way to

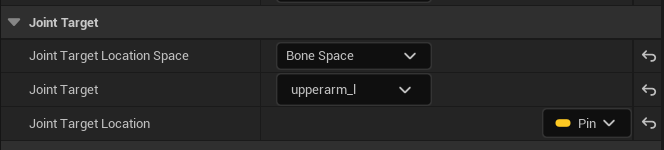
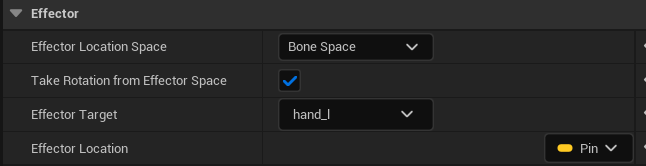
final joint coincides with that location as best it can.

4.leftHandIK things:

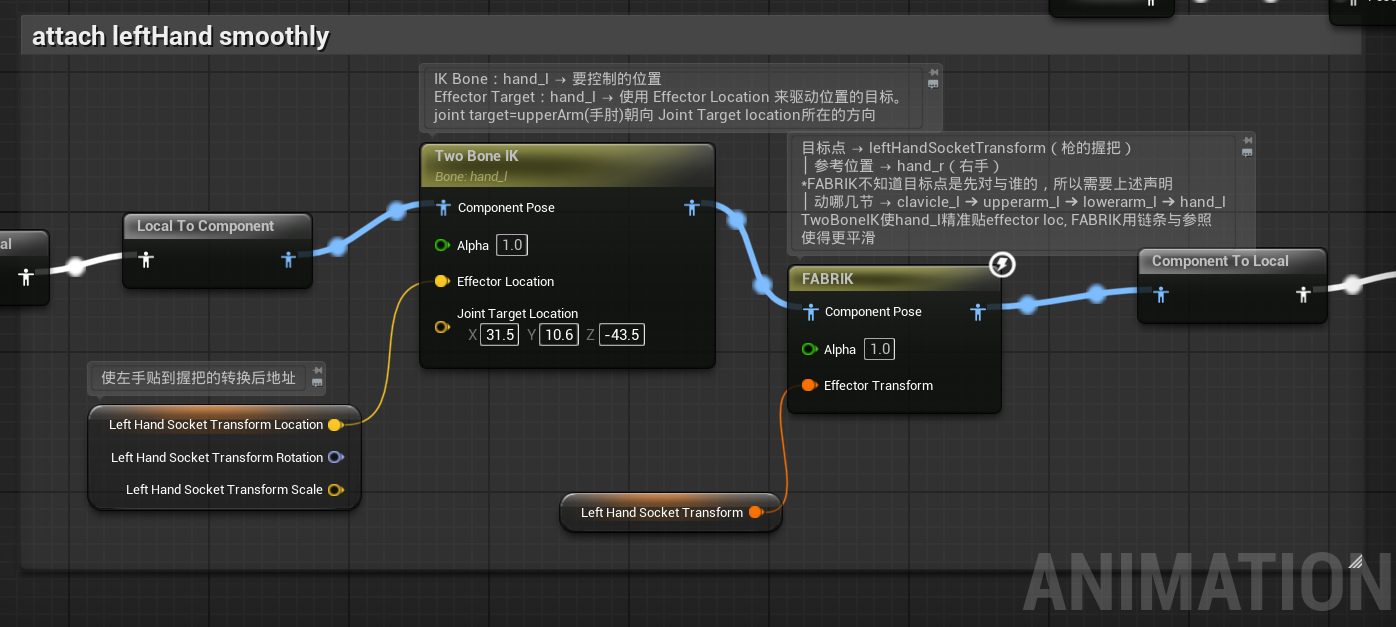
4.1 prepare work; in animGraph, add **two bone IK** and **FABRIK** node,

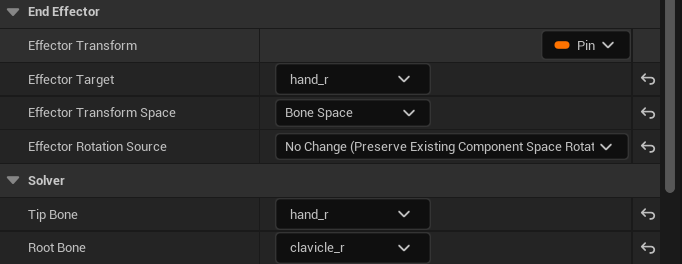
promoted a  **effector transform** variable named “**leftHandSocketTransform”**

4.2 



adjust default Joint TargetLocation when selected Two bone IK node

**FINAL EXPLAINATION**

4.3 in FABRIK

4.4 BluePrint Interface: more performance friendly way to calling variables from other blueprint

after create one, add new Function named IF\_getTransform, new return **Transform** variable

4.5 go to BP\_tppc, class settings to **implement interface**

and open showed function to edit it

4.5.1 create another BPI named **weaponInfo,**

implement inside of **weaponsBase blueprint,** make a function IF\_getWeaponTransform, return transform variable named “leftHandIKWorldTransform”

4.5.2 this to make weapons SK viable without cast



4.7 枪上的 leftHandIK Socket (世界空间)

↓

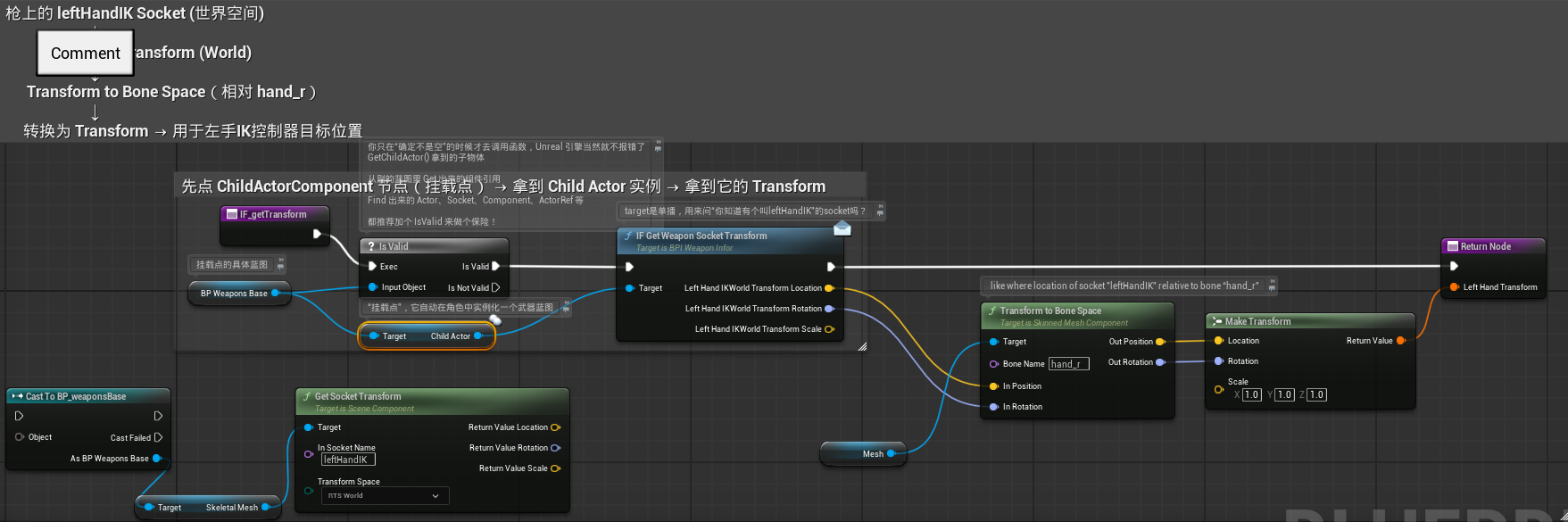
Get Socket Transform (World)

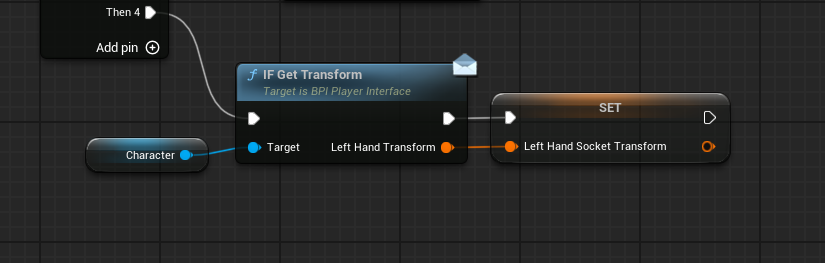
↓

Transform to Bone Space（相对 hand\_r）

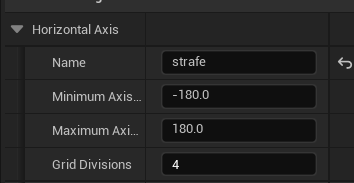
↓

转换为 Transform → 用于左手IK控制器目标位置

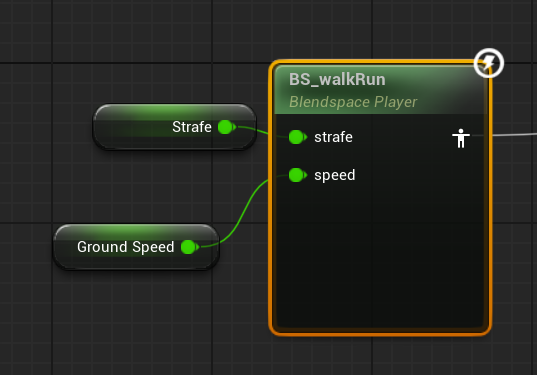


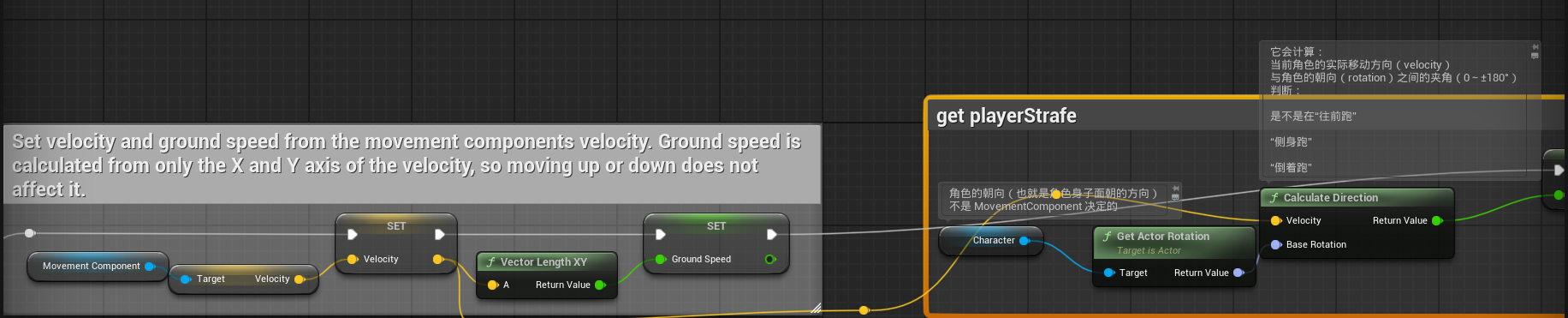
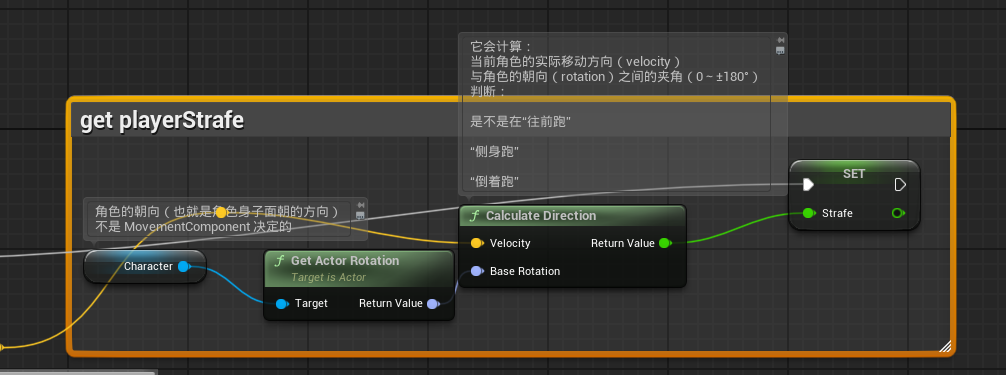
4.8 last: in abp\_manny:

p3. sprint, BS\_walkRun

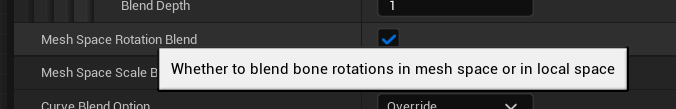
1.1 new blendSpace, 

leftTurn anim, rightTurn anim blah …

2.1 animBP->locomotion->walkRunpromote strafe, set it in eventGraph

2.2 in baseLocomotion:

3.1 change walk speed and other speed in BP\_TPC->characterMovement



enable it in locomotion->walkRun & Idle->layered blend per bone node

make animation sys more static, hands not sway like hell

4.1 create IA\_sprint and add it to IMC

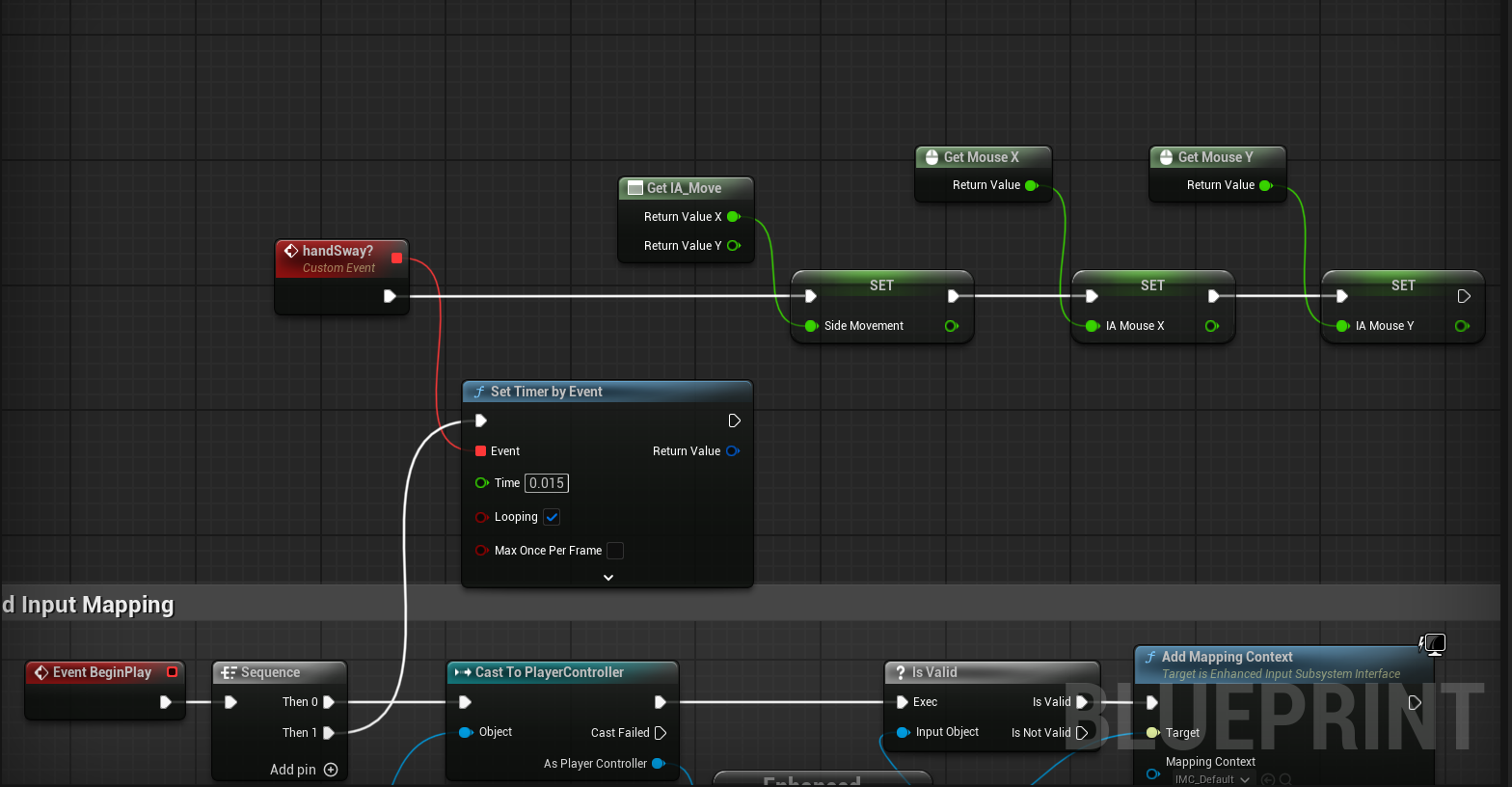
in BP\_TPC, implement this sprint logic via **get characterMovement->set walk speed.**

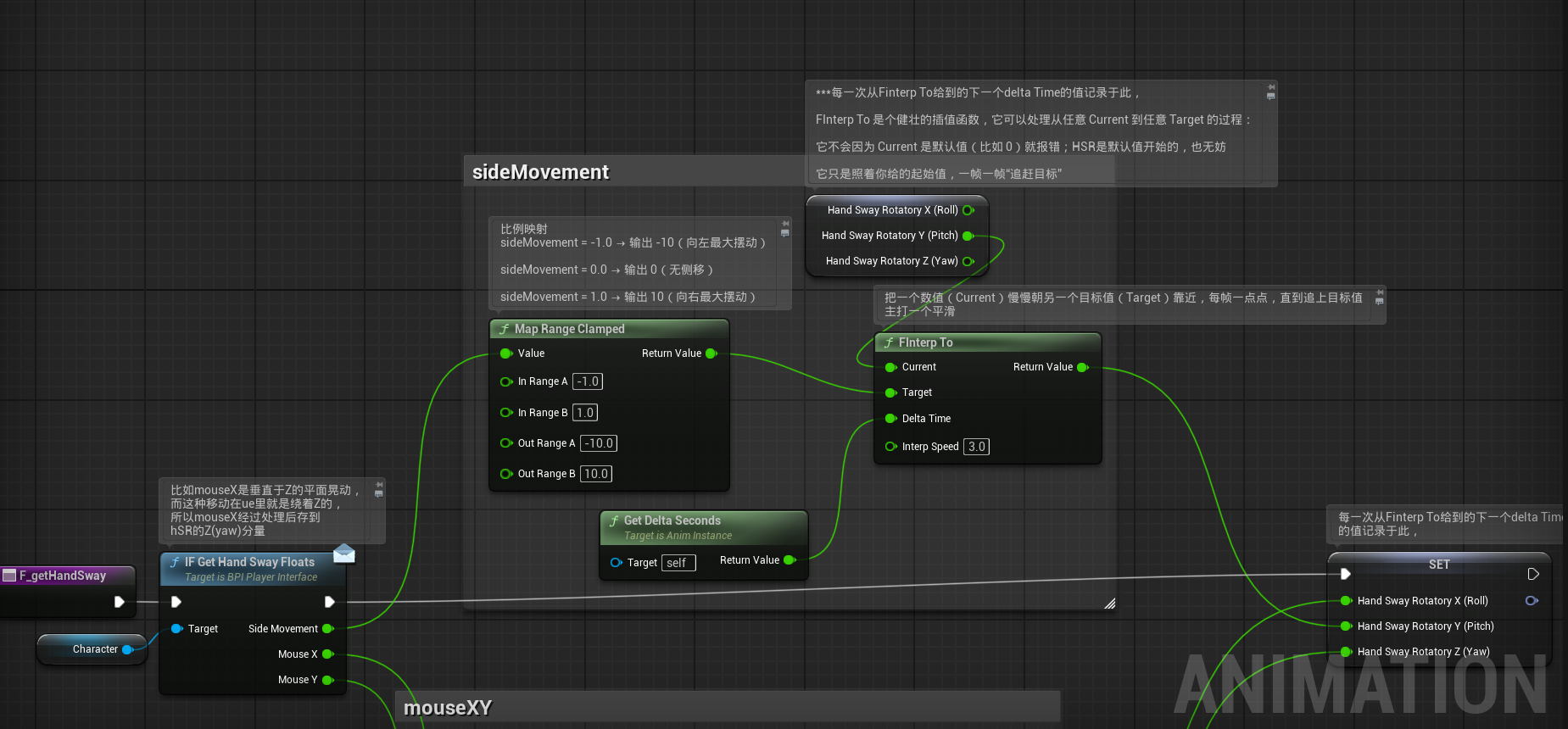
p4.handSway

1.1 in BPI\_playerInterface, add function IF\_getHandSway, three return value: float: sideMovement, mouseX, mouseY;

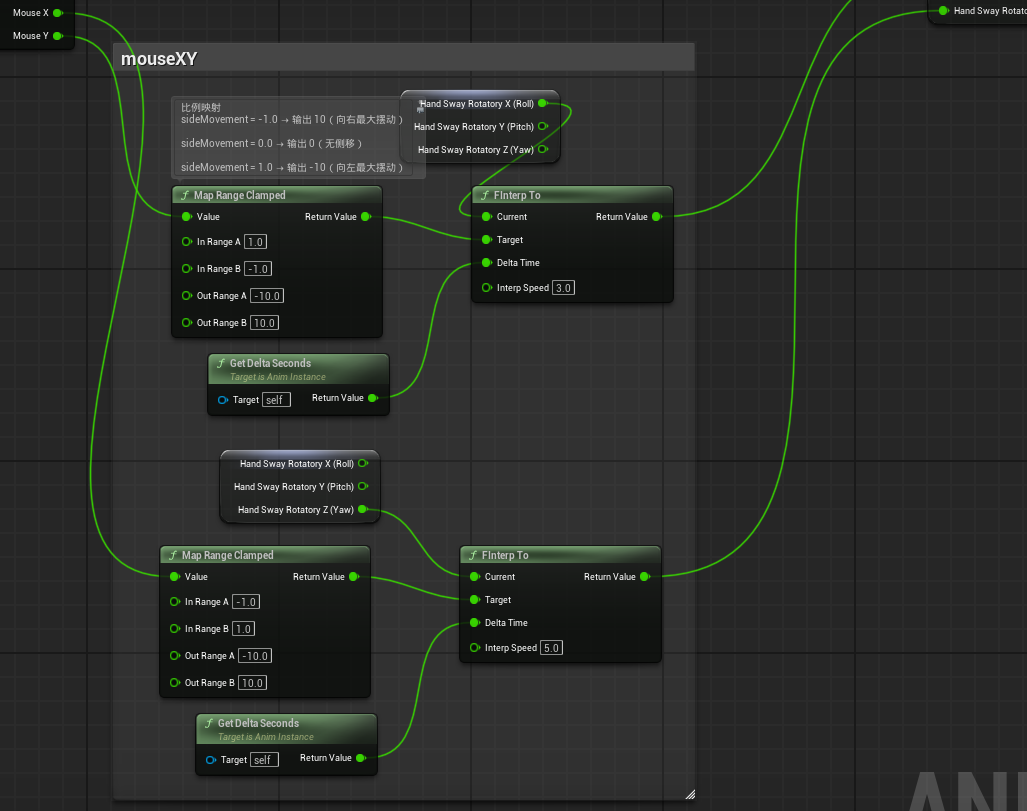
\*promote them of course, inside the BP\_TPC->IF\_getHandSway for sure;

\*category them into handSway;

1.2 set update those variable event on BP\_TPC:

1.3 in ABP\_manny, processing them like this logic in eventGraph:  


1.3.1 full view:



1.4: in animGraph: