目录

Virtual Game - Ryze

1. 四方向（程序化的八方向）

四方向所使用的枚举，具体演示将在 V2 推出

```
/** Types of virtual motion cardinal direction */
UENUM(BlueprintType)
namespace EMotionCardinalDirection
{
    enum Type
    {
        Forward,
        Backward,
        Left,
        Right,
        NONE,
    };
}
```

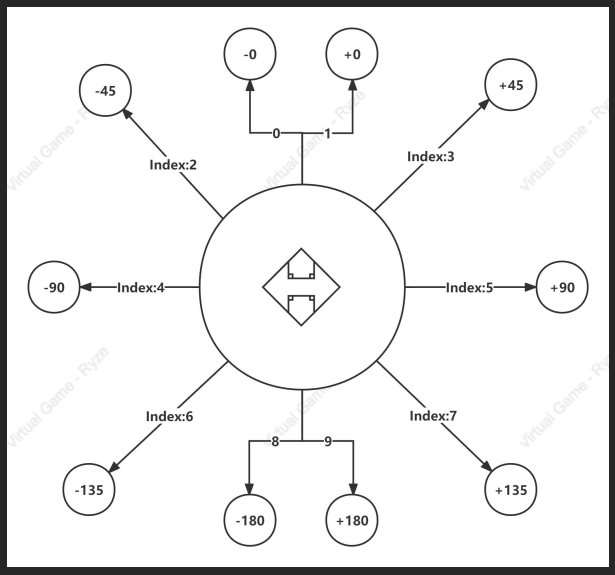
2. 十方向

（1）角度方向所使用的枚举

```
/** Types of virtual motion angle direction */
UENUM(BlueprintType)
namespace EMotionAngleDirection
{
    enum Type
    {
        ForwardLF,
        ForwardRF,
        ForwardLeft,
        ForwardRight,
        Left,
        Right,
        BackwardLeft,
        BackwardRight,
        BackwardLF,
        BackwardRF,
        NONE,
    };
}
```

Virtual Game - Ryze

（2）角度方向的概念图



（3）前方向和后方向都有左右脚区分（LF/RF），也就是 0 至-180 的逆时针旋转方向和 0 至+180 的顺时针旋转方向

3. 运动学状态

（1）动画状态所使用的枚举

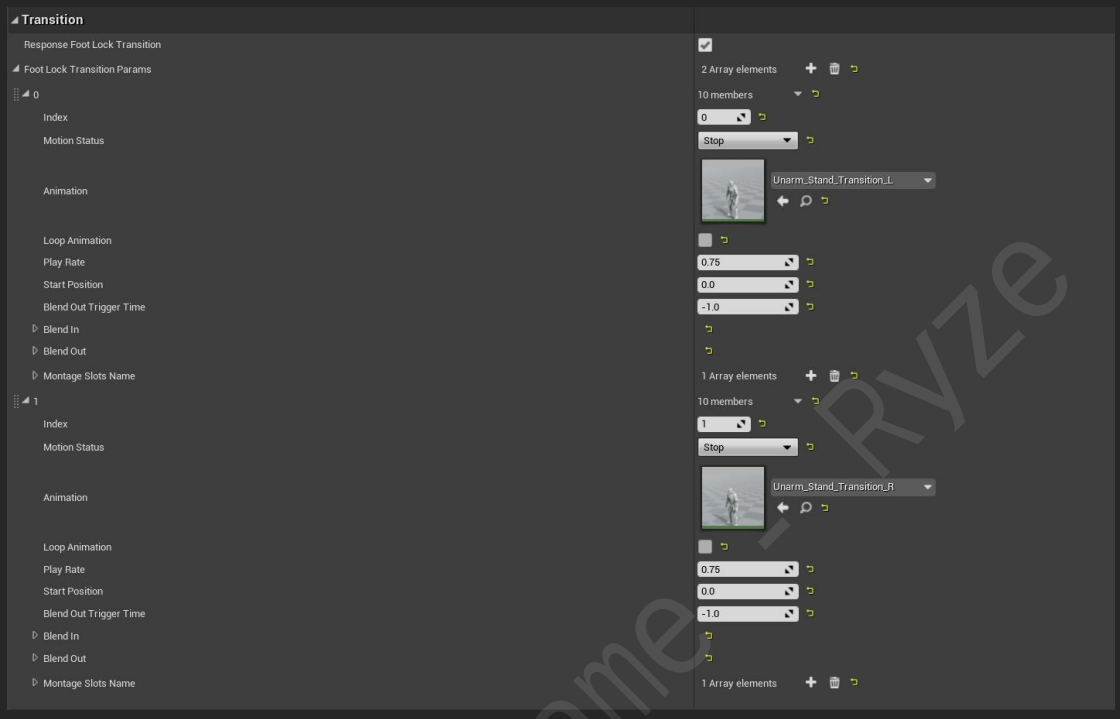
```
/** Types of virtual motion matching status */
UENUM(BlueprintType)
namespace EVirtualMotionStatus
{
    enum Type
    {
        Idle,
        Transition,
        TurnInPlace,
        Start,
        Moving,
        Pivot,
        Stop,
        JumpStart,
        JumpFalling,
        JumpLand,
        MAX UMETA(Hidden)
    };
}
```

Virtual Game - Ryze

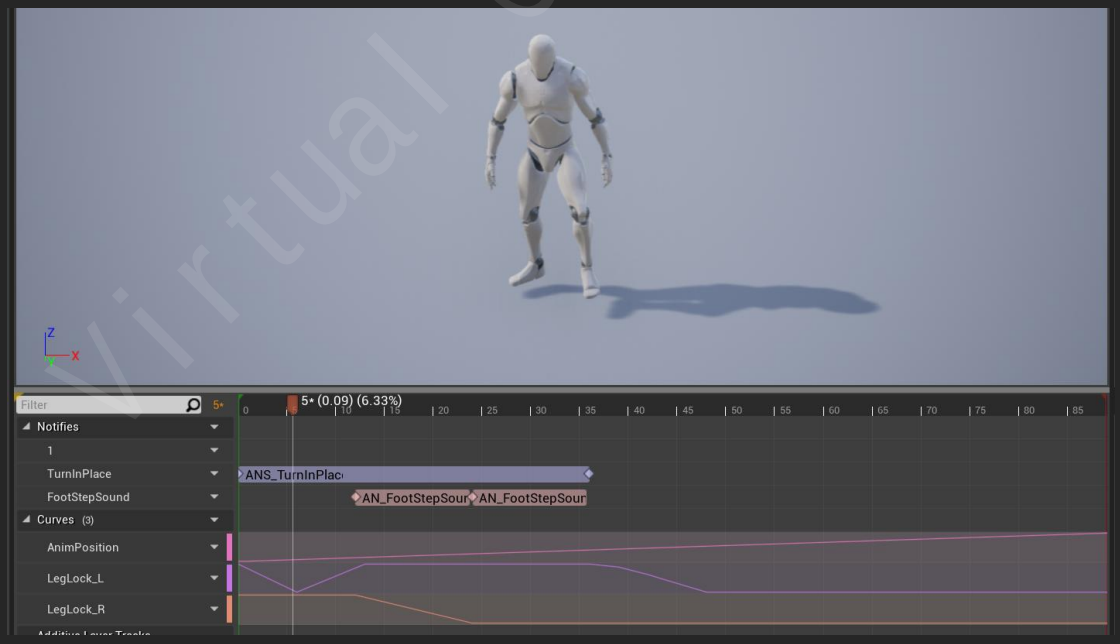
4. Transition 程序姿势过渡

（1）数组索引顺序对应的是左脚先起步[0]和右脚先起步[1]

（2）该功能主要是作为短位移的停步动画，以及作为各种姿势之间混合的程序过渡

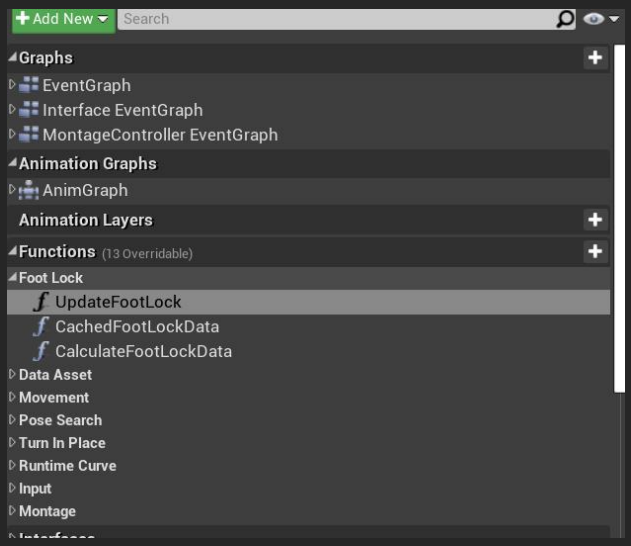


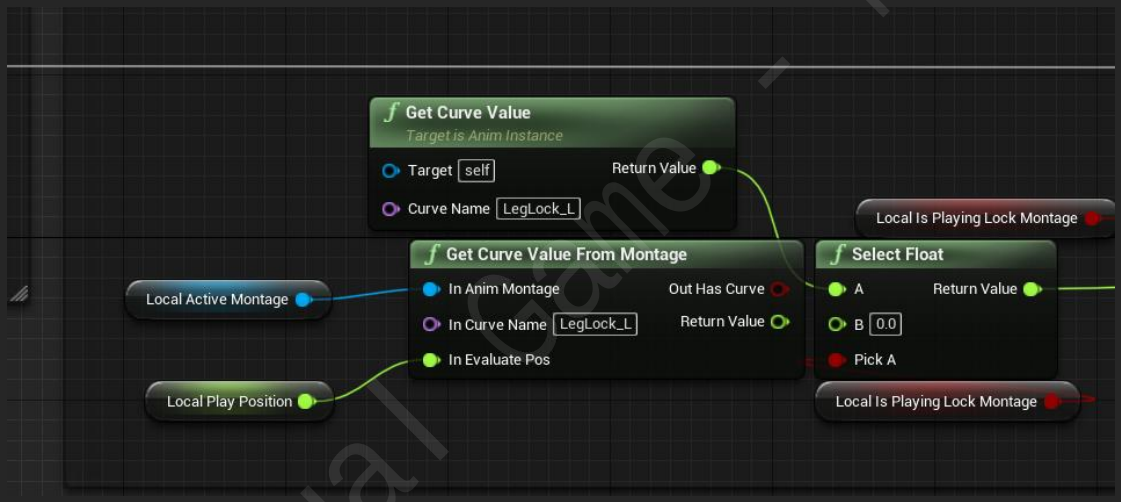
（3）你需要配置对应的 Leg Lock_L(R)锁定曲线



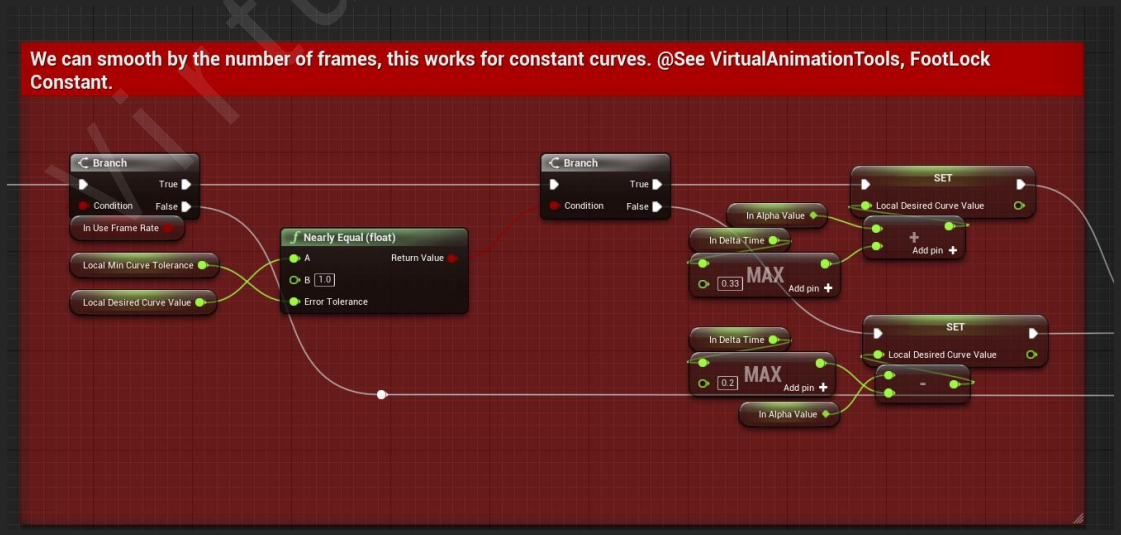
Virtual Game - Ryze

（4）我们可以选择获取曲线数据的模式（静态资产的曲线数据、实时混合的曲线数据）

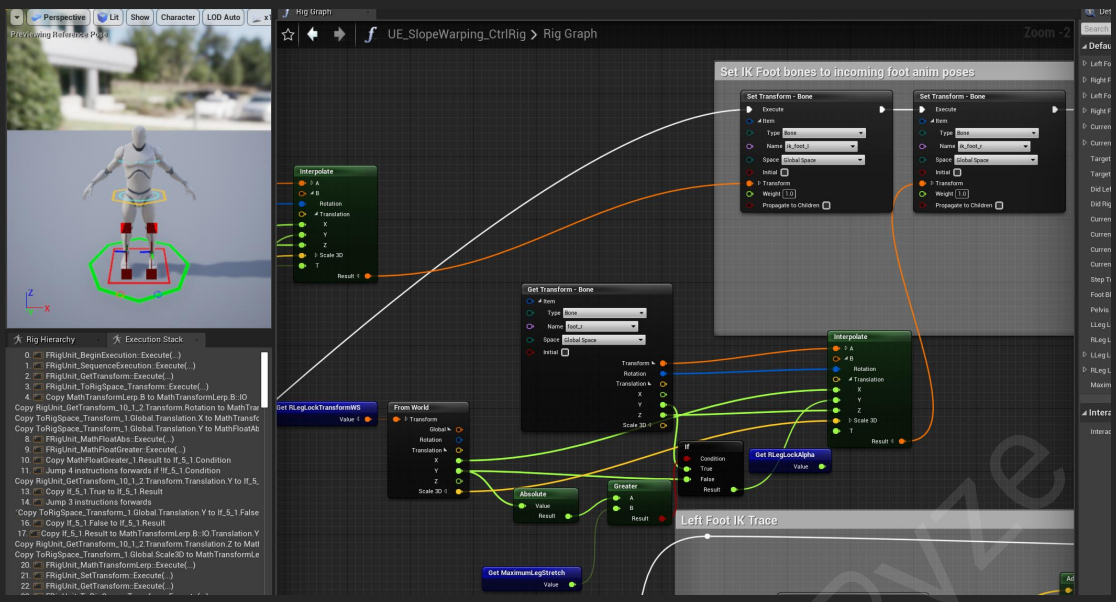




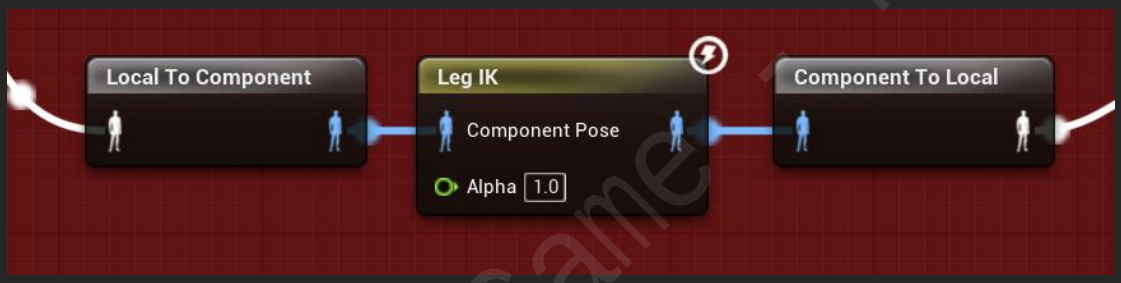
（5）帧过渡逻辑（避免因为数值出现骤降(升)的情况导致产生姿势顺变的效果）



（6）控制装置使用缓存的组件空间变换数据修改腿部数据（后续会推出一个新的腿部 IK 动画节点包含锁脚，距离匹配等）

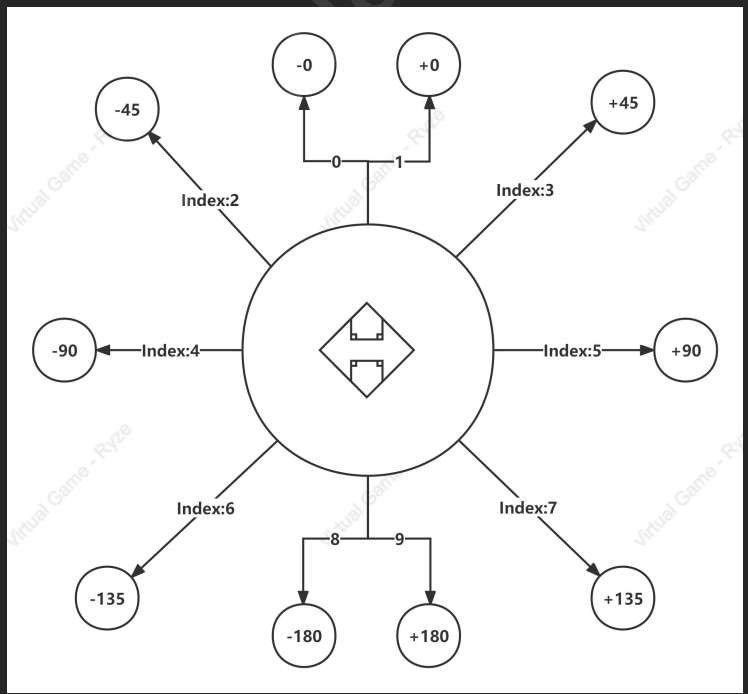


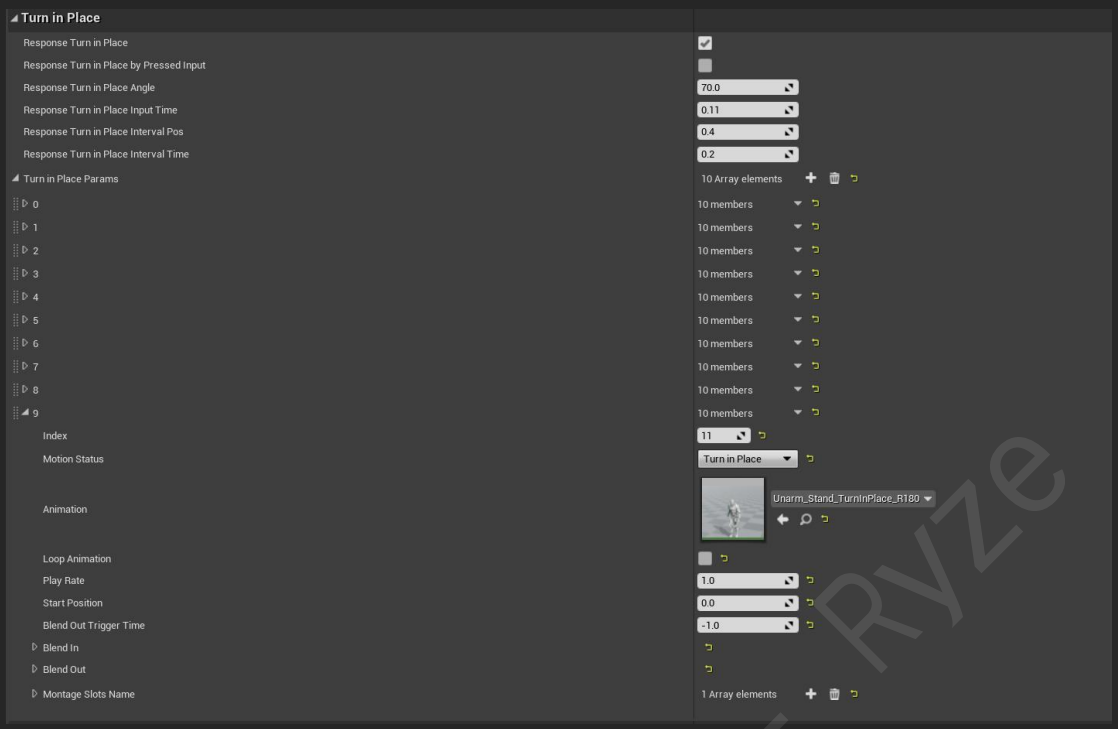
（7）使用腿部 IK 动画节点进行应用



5. Turn In Place 原地转身

（1）数组索引对应的是十方向





（2）参数

bResponseTurnInPlace：是否要播放转身动画

bResponseTurnInPlaceByPressedInput：响应转身的输入方式，按下输入键立即响应，或者是松开输入键后才响应

ResponseTurnInPlaceAngle：第一人称模式响应转身的角度范围（这里应该修改为左右区间）

ResponseTurnInPlaceInputTime：第一人称模式响应转身的等待时间

ResponseTurnInPlaceIntervalPos：如果正在播放转身动画，播放时间大于该值才可更新转身

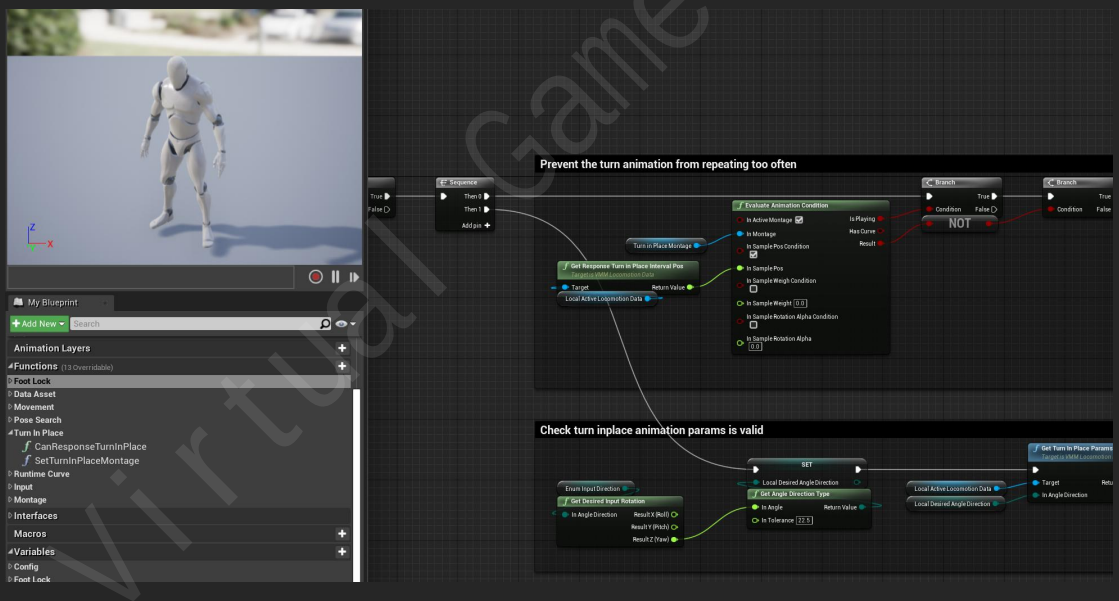
ResponseTurnInPlaceIntervalTime：如果正在播放转身动画，上一次播放的时间间隔大于该值才可更新转身

```
/** Defines whether to perform a turn animation in response */
UPROPERTY(EditAnywhere, BlueprintReadWrite, Category = TurnInPlace)
bool bResponseTurnInPlace;

/** If true, it will execute turn when the button is pressed, otherwise it will execute the released input */
UPROPERTY(EditAnywhere, BlueprintReadWrite, Category = TurnInPlace)
bool bResponseTurnInPlaceByPressedInput;

/** Minimum angle for responsive turn animation */
UPROPERTY(EditAnywhere, BlueprintReadWrite, Category = TurnInPlace)
float ResponseTurnInPlaceAngle;

/** If the time interval between start input and stop input is less than this value, we will play the turn animation */
UPROPERTY(EditAnywhere, BlueprintReadWrite, Category = TurnInPlace)
float ResponseTurnInPlaceInputTime;

/** If the turn animation is being played, check whether the current animation playback position is greater than this value, otherwise the ne
UPROPERTY(EditAnywhere, BlueprintReadWrite, Category = TurnInPlace)
float ResponseTurnInPlaceIntervalPos;

/** Minimum interval time for responding to turn animations, Only used in tick events, in order to get higher weight animation wait time */
UPROPERTY(EditAnywhere, BlueprintReadWrite, Category = TurnInPlace)
float ResponseTurnInPlaceIntervalTime;
```

Virtual Game - Ryze

```
    /** Return turn in place params from desired angle direction. */
    UFUNCTION(BlueprintCallable, Category = TurnInPlace, meta = (DisplayName = "Get Turn In Place Params"))
    FMotionAnimParamsData K2_GetTurnInPlaceParams(const EMotionAngleDirection::Type& InAngleDirection);
    const FMotionAnimParamsData* GetTurnInPlaceParams(const EMotionAngleDirection::Type& InAngleDirection);

public:

    /** Return the response turn in place animation condition */
    UFUNCTION(BlueprintPure, Category = TurnInPlace)
    FORCEINLINE bool CanResponseTurnInPlace() const { return bResponseTurnInPlace; }

    /** If true, it will execute turn when the button is pressed, otherwise it will execute the released input */
    UFUNCTION(BlueprintPure, Category = TurnInPlace)
    FORCEINLINE bool GetResponseTurnInPlacePressedState() const { return bResponseTurnInPlaceByPressedInput; }

    /** Return the response turn in place animation minimum angle */
    UFUNCTION(BlueprintPure, Category = TurnInPlace)
    FORCEINLINE float GetResponseTurnInPlaceAngle() const { return ResponseTurnInPlaceAngle; }

    /** Return the response turn in place animation input time */
    UFUNCTION(BlueprintPure, Category = TurnInPlace)
    FORCEINLINE float GetResponseTurnInPlaceInputTime() const { return ResponseTurnInPlaceInputTime; }

    /** Return the response turn in place animation interval position */
    UFUNCTION(BlueprintPure, Category = TurnInPlace)
    FORCEINLINE float GetResponseTurnInPlaceIntervalPos() const { return ResponseTurnInPlaceIntervalPos; }

    /** Return the response turn in place animation minimum interval time */
    UFUNCTION(BlueprintPure, Category = TurnInPlace)
    FORCEINLINE float GetResponseTurnInPlaceIntervalTime() const { return ResponseTurnInPlaceIntervalTime; }
```

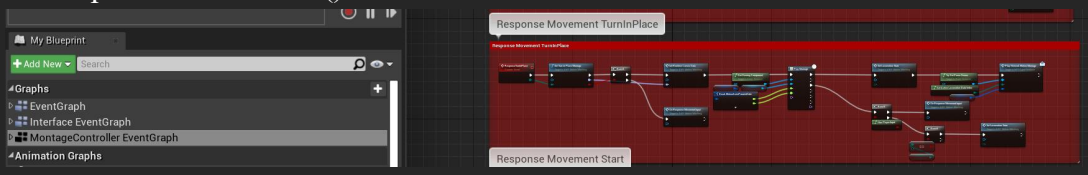
（3）功能



a. CanResponseTurnInPlace()：判断是否可以响应转身以及检测当前期望的转身动画数据是否有效

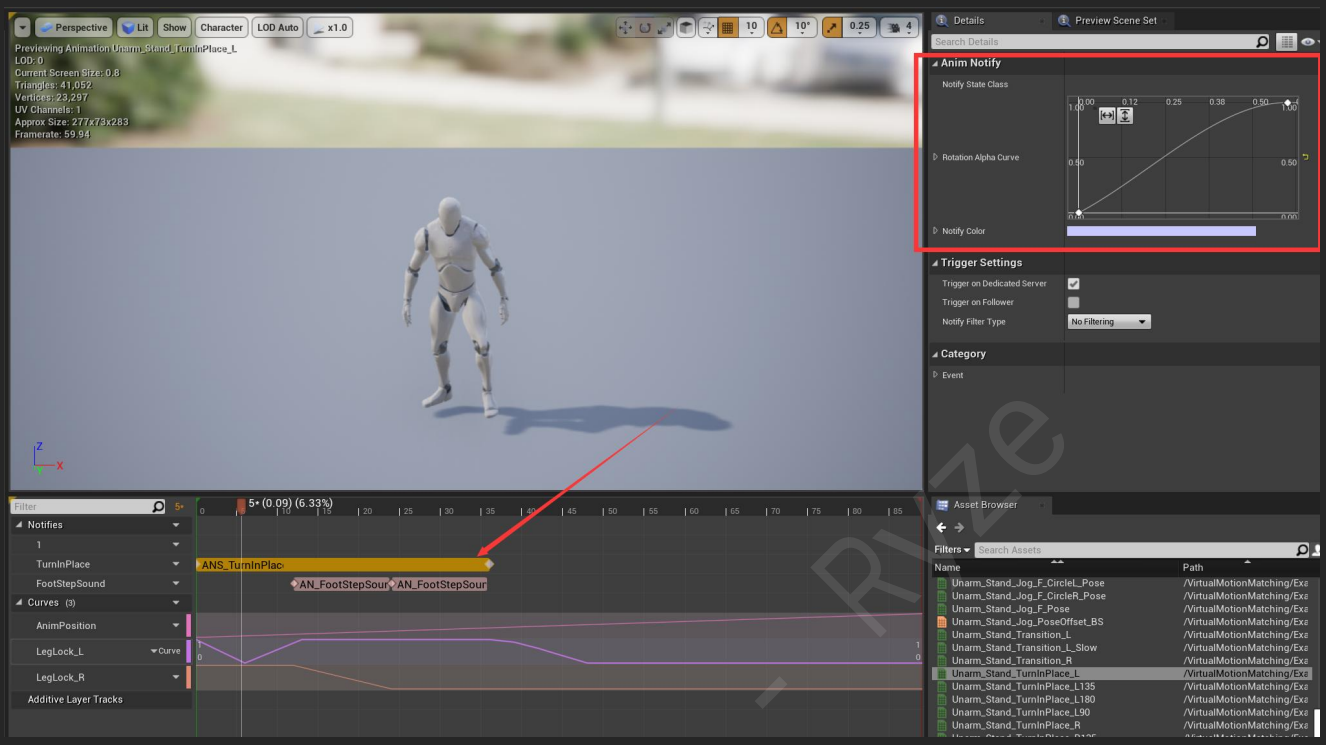
b. SetTurnInPlaceMontage()：缓存当前应该播放的转身动画

c. ResponseTurnInPlace()：执行播放转身动画



Virtual Game - Ryze

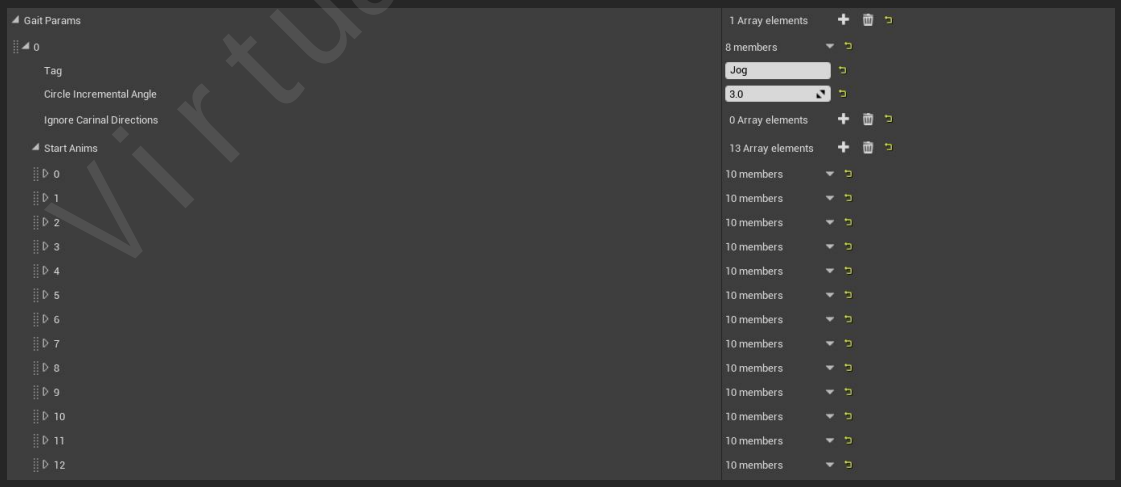
（4）程序动画曲线旋转（简单应用 0-1 的 Alpha 旋转曲线）



6. Movement Start 移动起步

（1）数组索引对应的是**十方向+三方向（Backwrd/Left/Right）**

0 - 9 即十方向索引　　10:Backward Start　　11:Left Start　　12:Right Start



Virtual Game - Rize

（2）参数

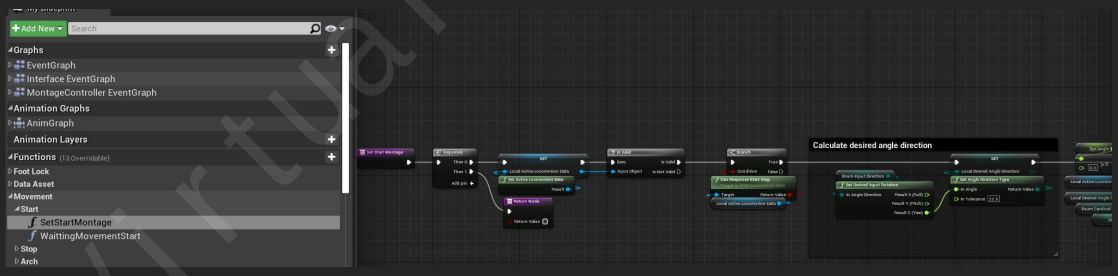
bResponseStartStep：是否要播放移动起步动画

```
#pragma region Start

protected:

    /** Defines whether to perform a start animation in response */
    UPROPERTY(EditAnywhere, BlueprintReadWrite, Category = Start)
    bool bResponseStartStep;

public:

    /** Return start animation params from desired data. */
    UFUNCTION(BlueprintCallable, Category = Start, meta = (DisplayName = "Get Start Params"))
    FMotionAnimParamsData K2_GetStartParams(bool InRightFoot, EMotionAngleDirection::Type InAngleDirection, EMoti
    const FMotionAnimParamsData* GetStartParams(bool InRightFoot, EMotionAngleDirection::Type InAngleDirection, E

public:

    /** Return the response start step animation condition */
    UFUNCTION(BlueprintPure, Category = Start)
    FORCEINLINE bool CanResponseStartStep() const { return bResponseStartStep; }

#pragma endregion
```

数据存储于 GaitParams

```
#pragma region Animation

protected:

    /** Idle animations. */
    UPROPERTY(EditAnywhere, BlueprintReadWrite, Category = Animation)
    UAnimSequence* BasePose;

    /** Gait animation in place, we can customize the index, Start, Moving, Stop, Pivot, Jump etc.. */
    UPROPERTY(EditAnywhere, BlueprintReadWrite, Category = Animation)
    TArray<FMotionGaitAnimsData> GaitParams;
```

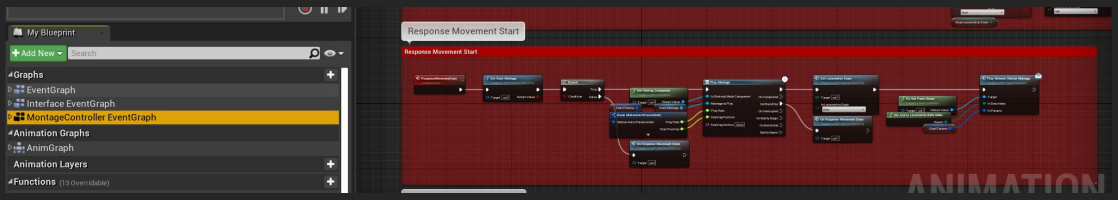
（3）功能



a. SetStartMontage()：缓存当前期望的移动起步动画

b. WaitingMovementStart()：等待响应播放移动起步，通常为了等待其它状态结束

c. ResponseMovementStart()：执行播放移动起步动画

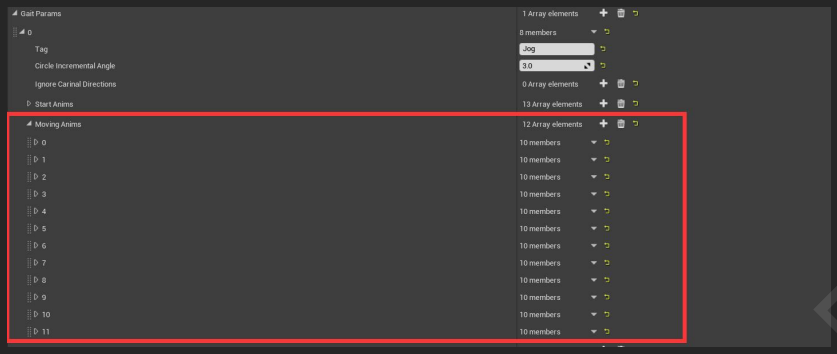


Virtual Game - Ryze

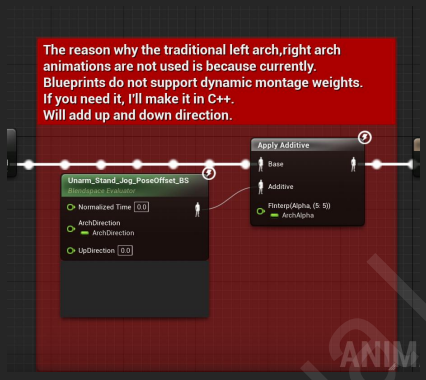
7. Movement Moving 移动循环

（1）数组索引对应的是四方向+左右圆弧转身

0:Forward 1:Forward Circle L 2:Forward Circle R
3:Backward 4: Backward Circle L 5.Backward Circle R
6:Left 7:Left Circle L 8:Left Circle R
9:Right 10:Right Circle L 11:Right Circle R



（如果没有对应动画，我们将使用程序偏移叠加资产，这里的偏移是使用 Circle L/R 的第一帧作为倾斜姿势，然后使用 VAT 骨骼混合层工具将手、脚、头与前方向姿势进行匹配）



（2）参数

CircleIncrementalAngle：每帧的旋转增量，该值参考弧形走动画的旋转帧差异曲线（使用 VAT 生成）

```
/** Struct of locomotion gait animations data */
USTRUCT(BlueprintType)
struct FMotionGaitAnimsData
{
    GENERATED_USTRUCT_BODY()

    /** Description for the current gait animations */
    UPROPERTY(EditAnywhere, BlueprintReadWrite)
    FName Tag;

    /** The angle by which the Circle rotates per frame */
    UPROPERTY(EditAnywhere, BlueprintReadWrite)
    float CircleIncrementalAngle;

    /** Whether to ignore the specified movement direction, it will be an invalid asset */
    UPROPERTY(EditAnywhere, BlueprintReadWrite)
    TArray<TEnumAsByte<EMotionCardinalDirection::Type>> IgnoreCarinalDirections;

    /** Movement start animations, 0 - 9 is forward start animation, 10 - 12 is other cardinal direction */
    UPROPERTY(EditAnywhere, BlueprintReadWrite)
    TArray<FMotionAnimParamsData> StartAnims;

    /** Movement loop animations, cardinal direction * (moving + left arch + right arch) */
    UPROPERTY(EditAnywhere, BlueprintReadWrite)
    TArray<FMotionAnimParamsData> MovingAnims;
```

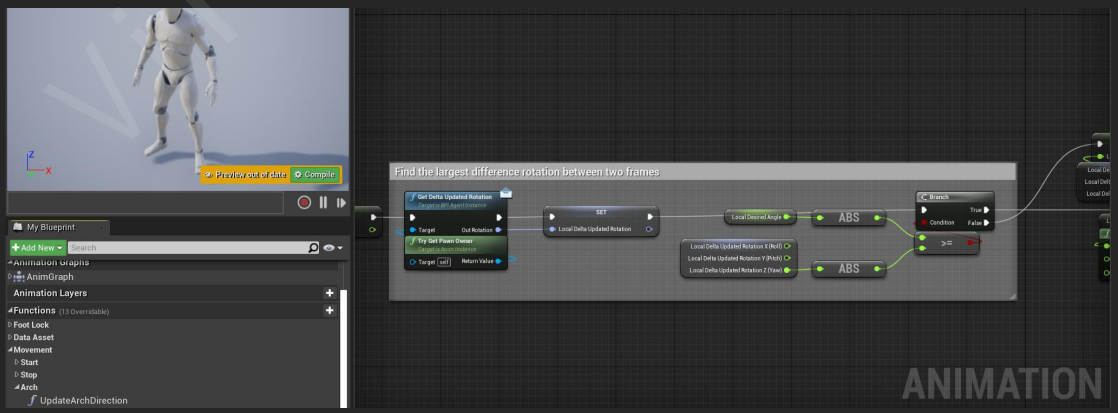
Virtual Game Ryze

数据存储于 GaitParams

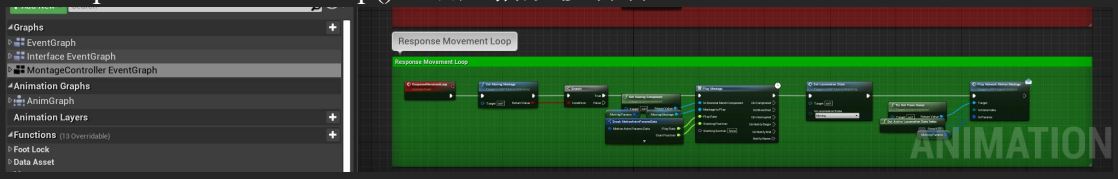
```
#pragma region Animation

protected:

    /** Idle animations. */
    UPROPERTY(EditAnywhere, BlueprintReadWrite, Category = Animation)
    UAnimSequence* BasePose;

    /** Gait animation in place, we can customize the index, Start, Moving, Stop, Pivot, Jump etc.. */
    UPROPERTY(EditAnywhere, BlueprintReadWrite, Category = Animation)
    TArray<FMotionGaitAnimsData> GaitParams;
```

（3）功能

a. UpdateArchDirection()：获取当前的转向值，该值是由控制器旋转减去当前角色正方向旋转所得的差异角度。并且将该角度与上一帧所旋转的值进行比较得出最大的旋转角度。



b. ResponseMovementLoop()：响应播放移动动画



解答疑问：为什么不适应混合空间呢？

（1）为了精简化状态机

（2）为了获取根骨骼的运动数据，状态机的根骨骼无法在多人联网模式使用（事实上，可以从状态机获得根骨骼运动数据，但是需要修改引擎，如果你感兴趣，可以联系我进一步探讨）

（3）多方向的混合以及弧形移动的混合功能将在之后推出，因为它的代码量略多，并且是一体化流程（即全逻辑编程都在 C++执行），将在之后单独推出以便更好的解释它的工作原理

8. Movement Pivot 移动转身

数组索引对应的是前方向的十方向转身以及四方向转身

0-19：Forward To 0-180 + LF/RF

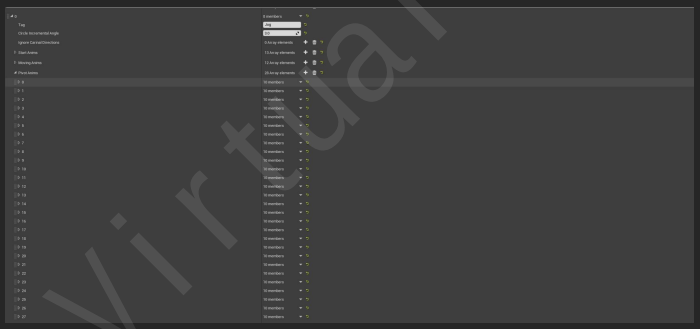
20-21：Forward To Backward + LF/RF

22-23：Backward To Forward + LF/RF

24-25：Left To Right + LF/RF

26-27：Right To Left + LF/RF

（如果没有对应动画，我们将使用 Start + Stop 模式模拟该移动转身效果）



（2）参数

bResponsePivotStep：是否响应移动转身

bResponseContinuousPivot：是否响应连续移动转身，如果是 True，可以在转身过程中继续转身

ResponsePivotAngle：响应移动转身的角度

ResponsePivotIntervalTime：响应移动转身的间隔时间

ResponsePivotByStartAnimPos：在起步状态中，起步动画播放位置大于等于该值才可以响应移动转身

ResponsePivotByStartAnimWeight：在起步状态中，起步动画播放权重大于等于该值才可以响应移动转身

Virtual Game - Ryze

ResponsePivotByStartAnimRotAlpha：在起步状态中，起步动画的旋转曲线权重大于等于该值才可以响应移动转身
ResponsePivotByStopAnimTime：在停步状态中，停步动画的播放位置大于等于该值才可以响应移动转身

```
    /** Defines whether to perform a pivot animation in response */
    UPROPERTY(EditAnywhere, BlueprintReadWrite, Category = Pivot)
    bool bResponsePivotStep;

    /** If it is true, the pivot animation will be played every time the condition for playing the pivot is reached, otherwise the Circle animati
    UPROPERTY(EditAnywhere, BlueprintReadWrite, Category = Pivot)
    bool bResponseContinuousPivot;

    /** Minimum angle(view + input) for responding to pivot animations */
    UPROPERTY(EditAnywhere, BlueprintReadWrite, Category = Pivot)
    float ResponsePivotAngle;

    /** Minimum interval for responding to pivot animations, Only used in tick events, in order to get higher weight animation wait time */
    UPROPERTY(EditAnywhere, BlueprintReadWrite, Category = Pivot)
    float ResponsePivotIntervalTime;

    /** If the start animation is being played, we can only execute pivot if the animation position is greater than this value. */
    UPROPERTY(EditAnywhere, BlueprintReadWrite, Category = Pivot)
    float ResponsePivotByStartAnimPos;

    /** If the start animation is being played, we can only execute pivot if the animation weight is greater than this value. */
    UPROPERTY(EditAnywhere, BlueprintReadWrite, Category = Pivot)
    float ResponsePivotByStartAnimWeight;

    /** If the start animation is being played, we can only execute pivot if the animation rotation alpha is greater than this value. */
    UPROPERTY(EditAnywhere, BlueprintReadWrite, Category = Pivot)
    float ResponsePivotByStartAnimRotAlpha;

    /**If the pivot animation is not played, but the stop-start pivot mode is used, we will stop the animation and execute the start animation af
    UPROPERTY(EditAnywhere, BlueprintReadWrite, Category = Pivot)
    float ResponsePivotByStopAnimTime;
```

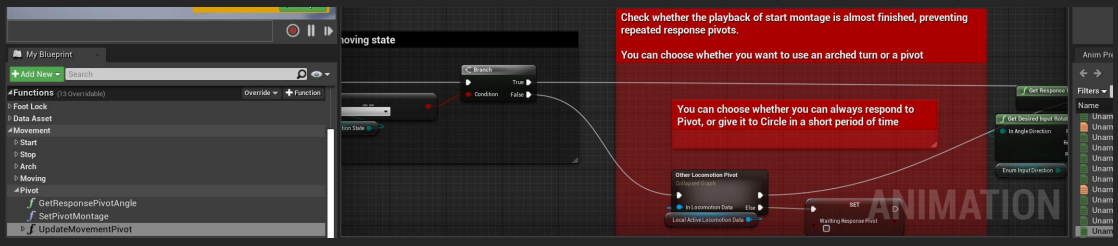
```
public:

    /** Return the response pivot animation condition */
    UFUNCTION(BlueprintPure, Category = Pivot)
    FORCEINLINE bool CanResponsePivotStep() const { return bResponsePivotStep; }

    /** Return the response continuous pivot animation condition */
    UFUNCTION(BlueprintPure, Category = Pivot)
    FORCEINLINE bool CanResponseContinuousPivot() const { return bResponseContinuousPivot; }

    /** Return the response pivot animation minimum angle */
    UFUNCTION(BlueprintPure, Category = Pivot)
    FORCEINLINE float GetResponsePivotAngle() const { return ResponsePivotAngle; }

    /** Return the response pivot animation minimum interval time */
    UFUNCTION(BlueprintPure, Category = Pivot)
    FORCEINLINE float GetResponsePivotIntervalTime() const { return ResponsePivotIntervalTime; }

    /** Return the response pivot animation by start animation position */
    UFUNCTION(BlueprintPure, Category = Pivot)
    FORCEINLINE float GetResponsePivotByStartAnimPos() const { return ResponsePivotByStartAnimPos; }

    /** Return the response pivot animation by start animation weight */
    UFUNCTION(BlueprintPure, Category = Pivot)
    FORCEINLINE float GetResponsePivotByStartAnimWeight() const { return ResponsePivotByStartAnimWeight; }

    /** Return the response pivot animation by start animation rotation alpha */
    UFUNCTION(BlueprintPure, Category = Pivot)
    FORCEINLINE float GetResponsePivotByStartAnimRotAlpha() const { return ResponsePivotByStartAnimRotAlpha; }

    /**If the pivot animation is not played, but the stop-start pivot mode is used, we will stop the animation and execute the start animation af
    UFUNCTION(BlueprintPure, Category = Pivot)
    FORCEINLINE float GetResponsePivotByStopAnimTime() const { return ResponsePivotByStopAnimTime; }
```

数据存储于 GaitParams

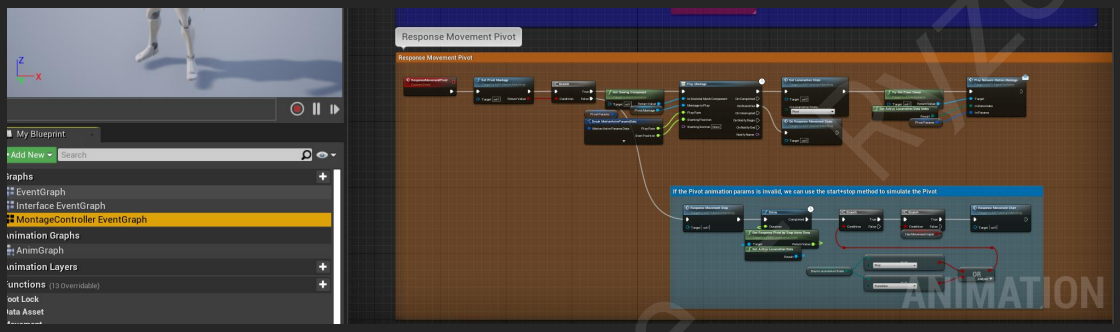
```
#pragma region Animation

protected:

    /** Idle animations. */
    UPROPERTY(EditAnywhere, BlueprintReadWrite, Category = Animation)
    UAnimSequence* BasePose;

    /** Gait animation in place, we can customize the index, Start, Moving, Stop, Pivot, Jump etc.. */
    UPROPERTY(EditAnywhere, BlueprintReadWrite, Category = Animation)
    TArray<FMotionGaitAnimsData> GaitParams;
```

Virtual Game - Ryze

（3）功能



a. GetResponsePivotAngle()：获取响应移动转身的角度
b. SetPivotMontage()：缓存期望的移动转身动画
c. UpdateMovementPivot()：每帧检测是否应该执行移动转身（当控制器旋转发生变化时）
d. ResponseMovementPivot()：响应播放移动转身动画



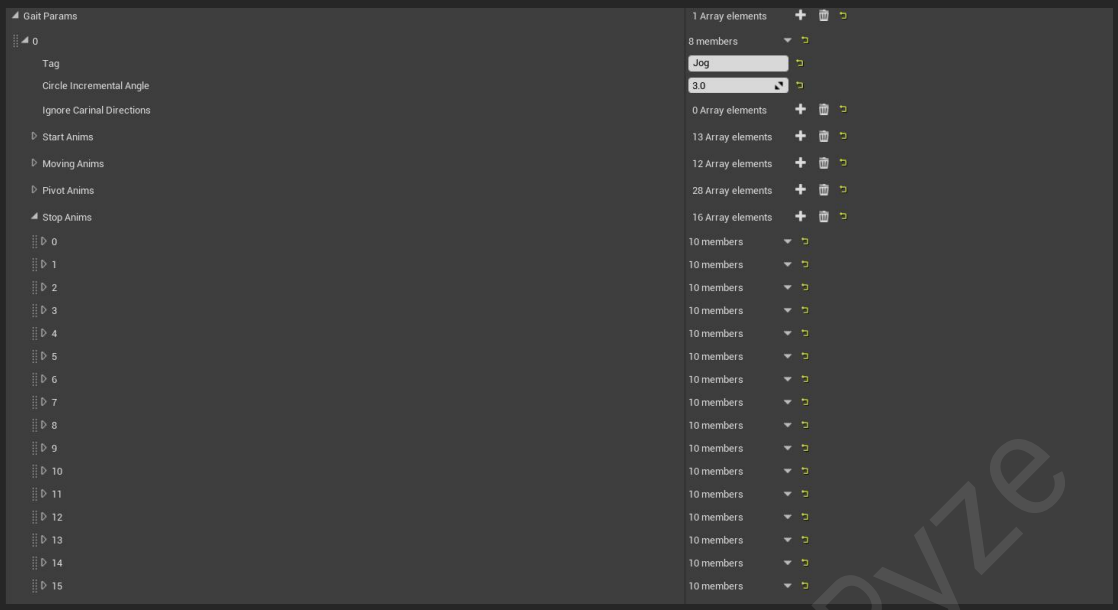
9. Movement Stop 移动停步

数组索引对应的是四方向左右脚停步*停步类型
0-3: Forward Stop LF/ Forward Stop RF/ Backward Stop LF/ Backward Stop RF
4-7: Left Stop LF/ Left Stop RF/ Right Stop LF/ Right Stop RF
每 8 个数据循环为一种 Stop Section 类型（类型：默认慢停步，快速急停）

```
/** Types of virtual motion stop section */
UENUM(BlueprintType)
namespace EMotionStopSection
{
    enum Type
    {
        Normal,
        Immediate,
        Max UMETA(Hidden)
    };
}
```

（V1 版本只需要配置前 4 个动画，Forward Stop LF/RF - Normal/Immediate）

Virtual Game Ryze



（2）参数

bResponseStopStep：是否响应移动停步

ResponseMovementInputIntervalTime：响应移动输入的间隔时间，可以防止短时间内重复播放动画

ResponseStopByStartAnimPos：在起步状态中，起步动画播放位置大于等于该值才可以响应停步

ResponseStopByStartAnimWeight：在起步状态中，起步动画播放权重大于等于该值才可以响应停步

ResponseStopByStartAnimRotAlpha：在起步状态中，起步动画旋转权重大于等于该值才可以响应停步

ResponseStopByPivotAnimPos：在移动转身状态中，移动转身动画播放位置大于等于该值才可以响应停步

ResponseStopByPivotAnimWeight：在移动转身状态中，移动转身动画播放权重大于等于该值才可以响应停步

ResponseStopByPivotAnimRotAlpha：在移动转身状态中，移动转身动画旋转权重大于等于该值才可以响应停步

```
    /** Return stop animation params from input data. */
    UFUNCTION(BlueprintCallable, Category = Moving, meta = (DisplayName = "Get Stop Params"))
    FMotionAnimParamsData K2_GetStopParams(const int32 InWaitAxis, const int32 InFootIndex, const EMotionCardinalDirection::Type InDirectionType, EMotionStopSectio
    const FMotionAnimParamsData* GetStopParams(const int32& InWaitAxis, const int32& InFootIndex, const EMotionCardinalDirection::Type& InDirectionType, EMotionSto

public:

    /** Return the response pivot animation condition */
    UFUNCTION(BlueprintPure, Category = Stop)
    FORCEINLINE bool CanResponseStopStep() const { return bResponseStopStep; }

    /** Return the response movement input interval time from last stop record time */
    UFUNCTION(BlueprintPure, Category = Stop)
    FORCEINLINE float GetResponseMovementInputIntervalTime() const { return ResponseMovementInputIntervalTime; }

    /** Return the response stop animation by start animation position */
    UFUNCTION(BlueprintPure, Category = Stop)
    FORCEINLINE float GetResponseStopByStartAnimPos() const { return ResponseStopByStartAnimPos; }

    /** Return the response stop animation by start animation weight */
    UFUNCTION(BlueprintPure, Category = Stop)
    FORCEINLINE float GetResponseStopByStartAnimWeight() const { return ResponseStopByStartAnimWeight; }

    /** Return the response stop animation by start animation rotation alpha */
    UFUNCTION(BlueprintPure, Category = Stop)
    FORCEINLINE float GetResponseStopByStartAnimRotAlpha() const { return ResponseStopByStartAnimRotAlpha; }

    /** Return the response stop animation by pivot animation position */
    UFUNCTION(BlueprintPure, Category = Stop)
    FORCEINLINE float GetResponseStopByPivotAnimPos() const { return ResponseStopByPivotAnimPos; }

    /** Return the response stop animation by pivot animation weight */
    UFUNCTION(BlueprintPure, Category = Stop)
    FORCEINLINE float GetResponseStopByPivotAnimWeight() const { return ResponseStopByPivotAnimWeight; }

    /** Return the response stop animation by pivot animation rotation alpha */
    UFUNCTION(BlueprintPure, Category = Stop)
    FORCEINLINE float GetResponseStopByPivotAnimRotAlpha() const { return ResponseStopByPivotAnimRotAlpha; }
```

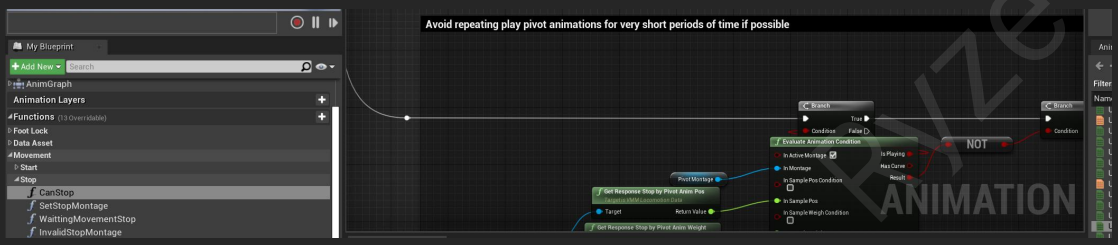
Virtual Camera - Ryze

数据存储于 GaitParams

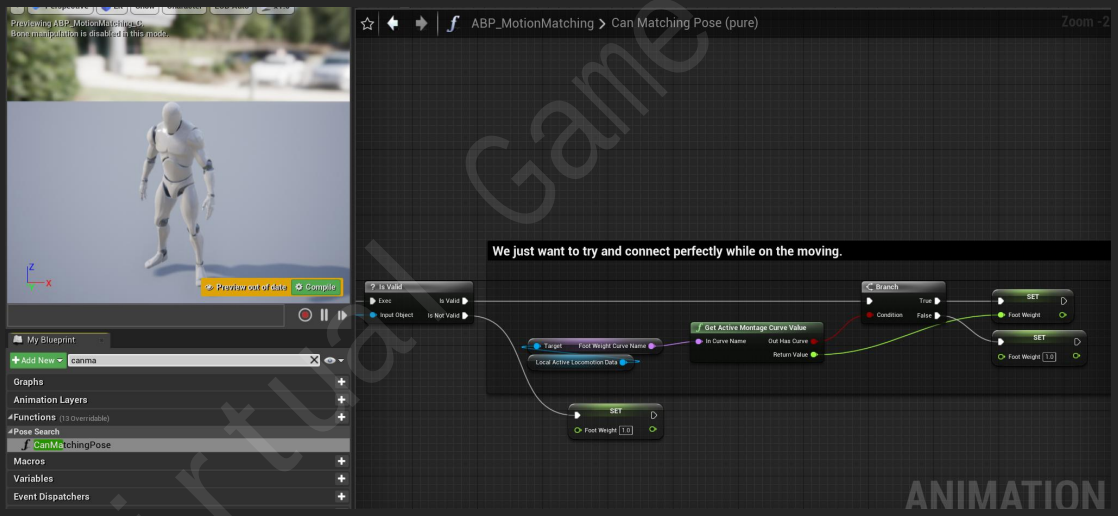
```
#pragma region Animation

protected:

    /** Idle animations. */
    UPROPERTY(EditAnywhere, BlueprintReadWrite, Category = Animation)
    UAnimSequence* BasePose;

    /** Gait animation in place, we can customize the index, Start, Moving, Stop, Pivot, Jump etc.. */
    UPROPERTY(EditAnywhere, BlueprintReadWrite, Category = Animation)
    TArray<FMotionGaitAnimsData> GaitParams;
```

（3）功能

a. CanStop()：返回是否可以执行停步



b. CanMatchingPose()：返回当前姿势是否匹配停步动画姿势（参考 VAT 脚步权重工具）

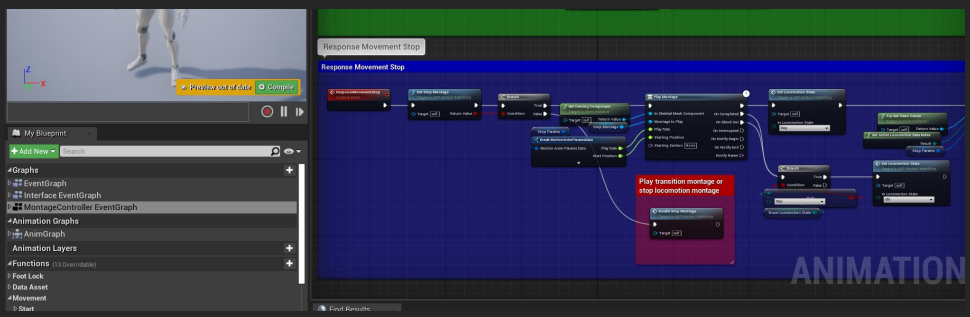


c. SetStopMontage()：缓存期望的停步动画

d. WaitingMovementStop()：等待响应停步（如果当前姿势不匹配等情况）

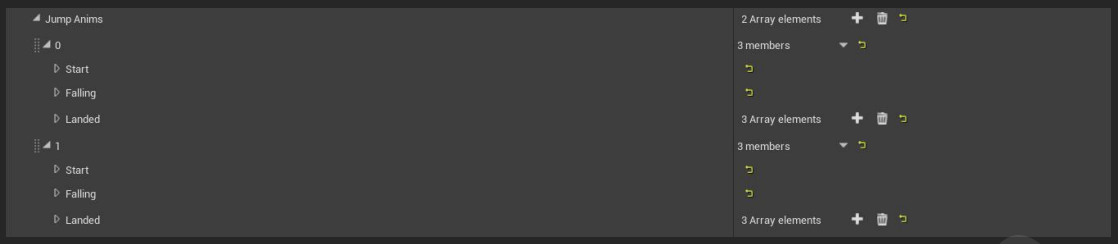
e. InvalidStopMontage()：如果是无效的停步动画（状态不匹配等情况）我们可以选择使用程序姿势过渡的方式停步或者直接停止运行中的动画

f. ResponseMovementStop()：响应播放停步动画



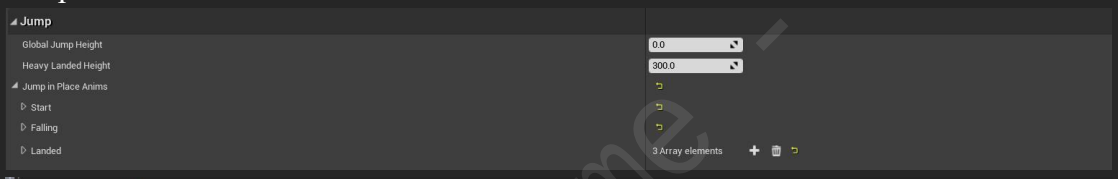
10. Jump 跳跃

数组索引对应的是前方向的十方向转身以及四方向转身
0-3: Forward Jump LF/ Forward Jump RF/ Backward Jump LF/ Backward Jump RF
4-7: Left Jump LF/ Left Jump RF/ Right Jump LF/ Right Jump RF



(V1 版本只需要配置前 2 个动画，Forward Jump LF/RF)

（2）参数
GlobalJumpHeight：全局跳跃高度，将在 V2 版本推出自定义跳跃高度的功能
HeavyLandedHeight：当坠落高度大于该值，我们将使用不同的落地动画
JumpInPlaceAnims：原地跳跃动画数据



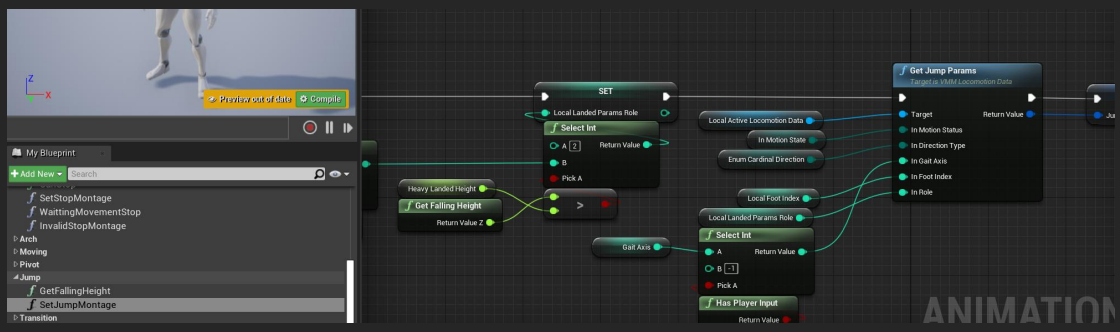

```cpp
#pragma region Jump

protected:

    /** The default global jump height */
    UPROPERTY(EditAnywhere, BlueprintReadWrite, Category = Jump)
    float GlobalJumpHeight;

    /** If the height of the fall is greater than this value, heavy landed will be executed */
    UPROPERTY(EditAnywhere, BlueprintReadWrite, Category = Jump)
    float HeavyLandedHeight;

    /** Movement jump InPlace animations */
    UPROPERTY(EditAnywhere, BlueprintReadWrite, Category = Jump)
    FMotionJumpAnimsData JumpInPlaceAnims;

public:

    /** Return jump animation params from input data. */
    UFUNCTION(BlueprintCallable, Category = Moving, meta = (DisplayName = "Get Jump Params"))
        FMotionAnimParamsData K2_GetJumpParams(const EVirtualMotionStatus::Type& InMotionStatus, const EMotionCardinalDirection::Type& InDirectionType
            , const int32& InGaitAxis, const int32& InFootIndex, const int32 InRole = 0);
    const FMotionAnimParamsData* GetJumpParams(const EVirtualMotionStatus::Type& InMotionStatus, const EMotionCardinalDirection::Type& InDirectionType
        , const int32& InGaitAxis, const int32& InFootIndex, const int32 InRole = 0);

public:

    /** Return the global jump height */
    UFUNCTION(BlueprintPure, Category = Jump)
    FORCEINLINE float GetGlobalJumpHeight() const { return GlobalJumpHeight; }

    /** Return the heavy landed height */
    UFUNCTION(BlueprintPure, Category = Jump)
    FORCEINLINE float GetHeavyLandedHeight() const { return HeavyLandedHeight; }

#pragma endregion
```

数据存储于 GaitParams

```cpp
#pragma region Animation

protected:

    /** Idle animations. */
    UPROPERTY(EditAnywhere, BlueprintReadWrite, Category = Animation)
    UAnimSequence* BasePose;

    /** Gait animation in place, we can customize the index, Start, Moving, Stop, Pivot, Jump etc.. */
    UPROPERTY(EditAnywhere, BlueprintReadWrite, Category = Animation)
    TArray<FMotionGaitAnimsData> GaitParams;
```

Virtual Game - Wize

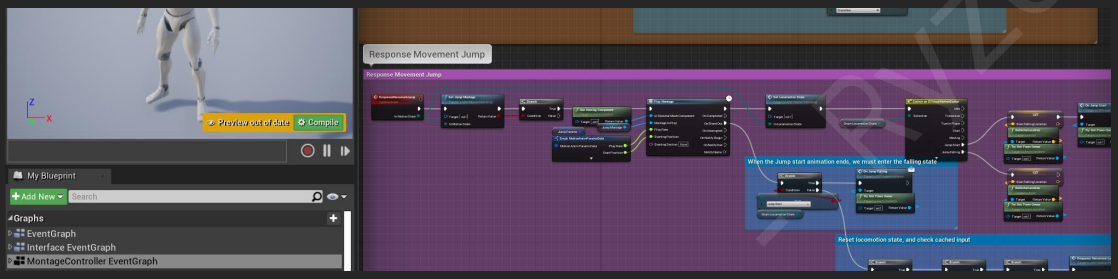
（3）功能



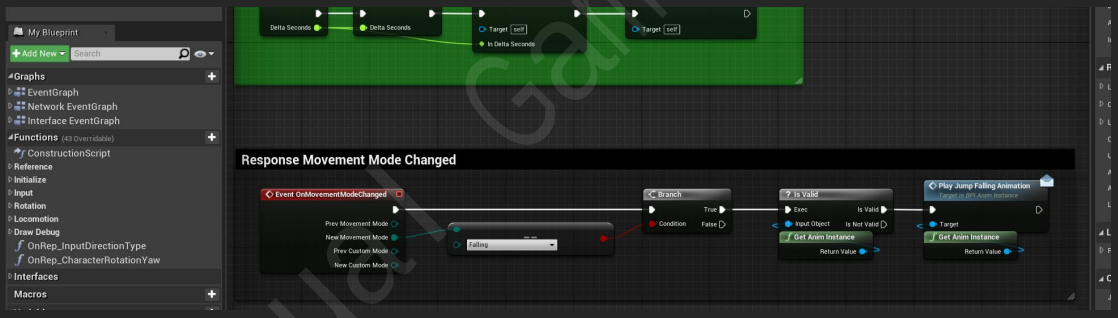
a. GetFallingHeight()：获取坠落高度

b. SetJumpMontage()：缓存期望的跳跃动画

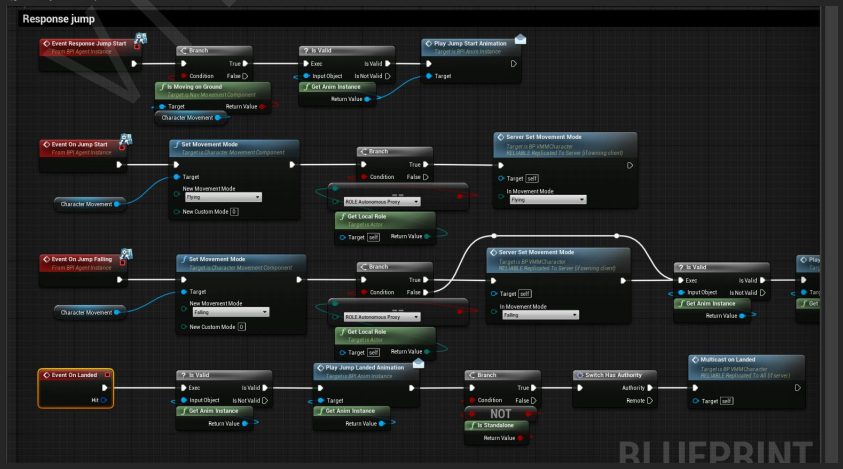
c. Response Movement Jump()：响应播放停步动画



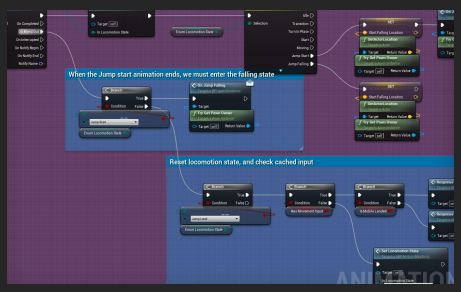
d. ResponseMovementModeChanged()：一旦移动模式切换为 Falling，我们应该通知动画蓝图播放对应的坠落动画



e. OnLanded()：一旦触发了落地回调，我们应该通知动画蓝图播放对应的落地动画，这里只有本地控制器和服务器才会触发，所以我们可以在服务器触发时通知模拟端进行落地

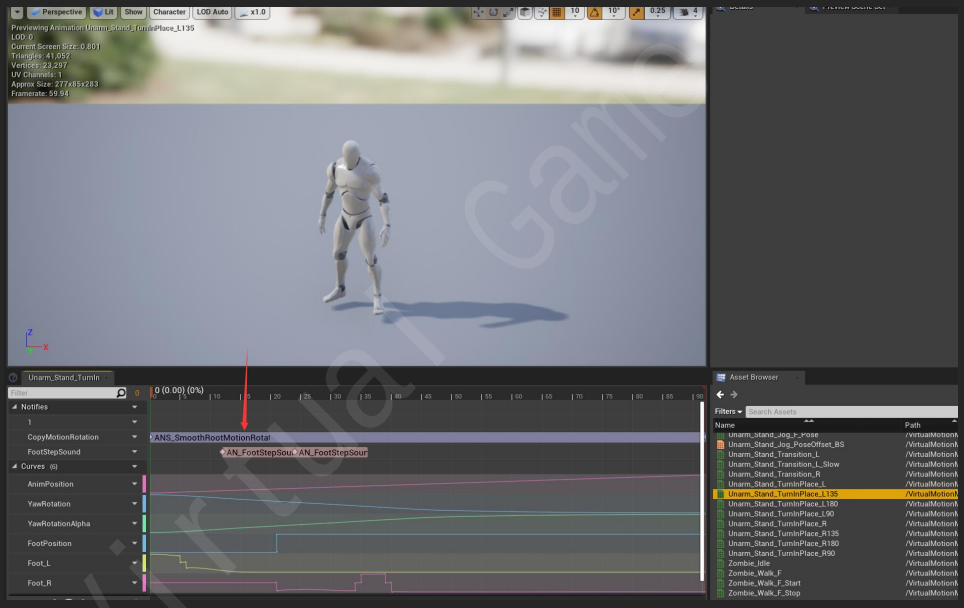


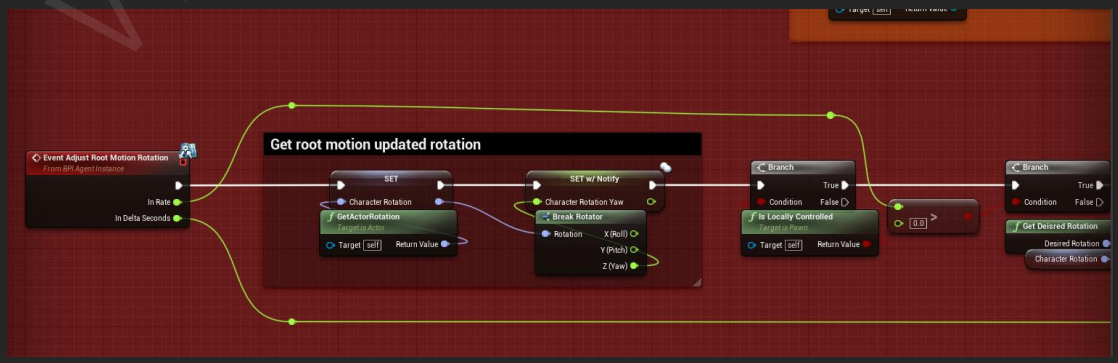
f. OnJumpStart()：当前的跳跃模式设置为飞行，在动画混出时设置为 Falling 模式，执行引擎的原流程



11. 根骨骼与自定义混合旋转

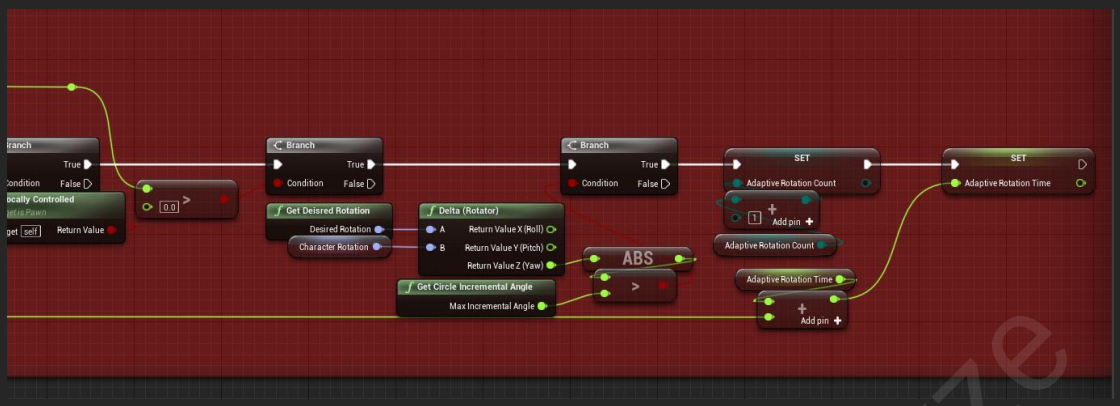
（1）我们需要在每一个带有旋转数据的根骨骼动画上配置 ANS_SmoothRootMotionRotation 动画通知状态（可以使用 VAT 工具批量配置）



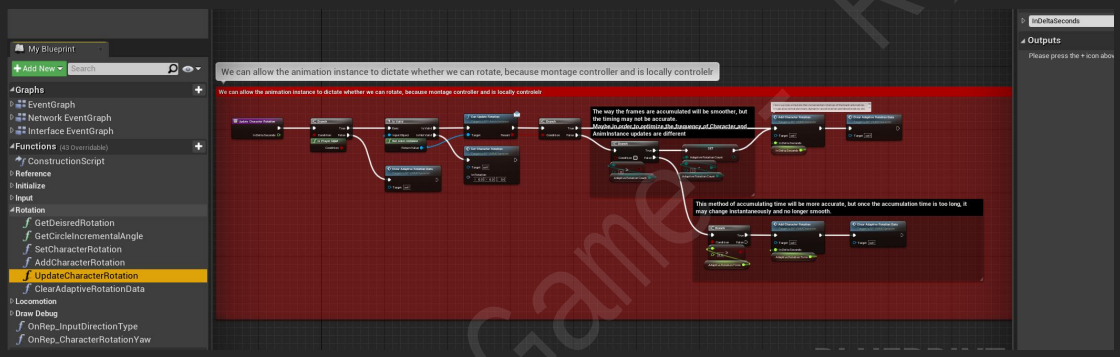


该动画通知会将当前的角色旋转缓存至自定义的 CharacterRotation 变量中，并在服务器上同步 Yaw 方向的数值

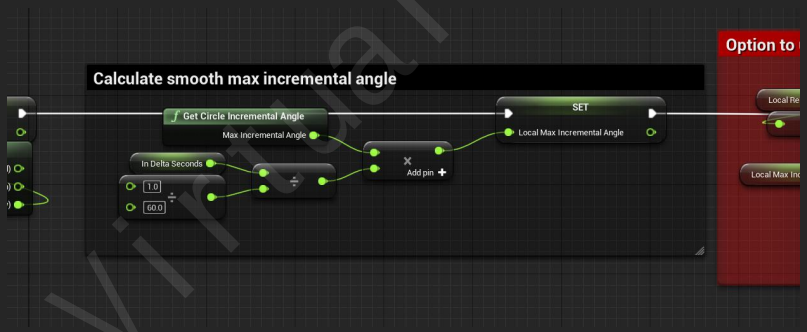
（2）如果该动画通知存在增量旋转的情况，我们应该延迟至下一帧执行旋转，这是因为组件更新的时序导致的问题。
Tick 更新顺序：CharacterMovement→Character→AnimInstance



（3）UpdateCharacterRotation 该函数负责管理角色旋转的更新



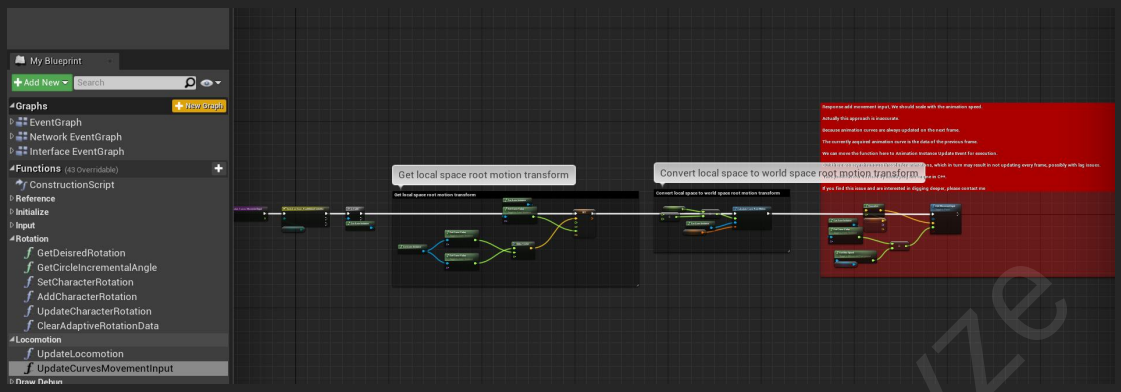
（4）AddCharacterRotation 该函数负责每帧的增量旋转（这里我们应该将每帧旋转角度与默认 60 帧的动画数据进行转换，保持不同帧率下旋转的数值是恒定的）



Virtual Game Ryze

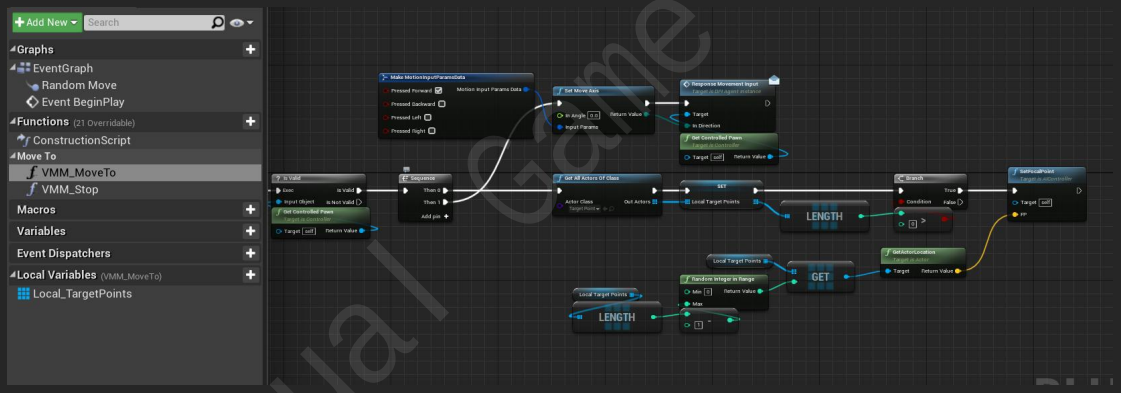
12. 曲线控制器

由于该方法存在误差且该插件已经实现了更精确的蒙太奇控制器，所以只有简单的代码框架（将在之后弃用）



13. AI

当前 AI 只有简单的移动停止行为，后续将有使用运动匹配的 3A-AI 完整案例



14. 脚步曲线

哔哩哔哩：

（1）脚步 IK：

https://www.bilibili.com/video/BV1744y1V7CS/?spm_id_from=333.788

（2）脚步锁定：

https://www.bilibili.com/video/BV1R3411J7h6/?spm_id_from=333.788

（3）脚步权重：

https://www.bilibili.com/video/BV1qT4y1Y7to/?spm_id_from=333.788

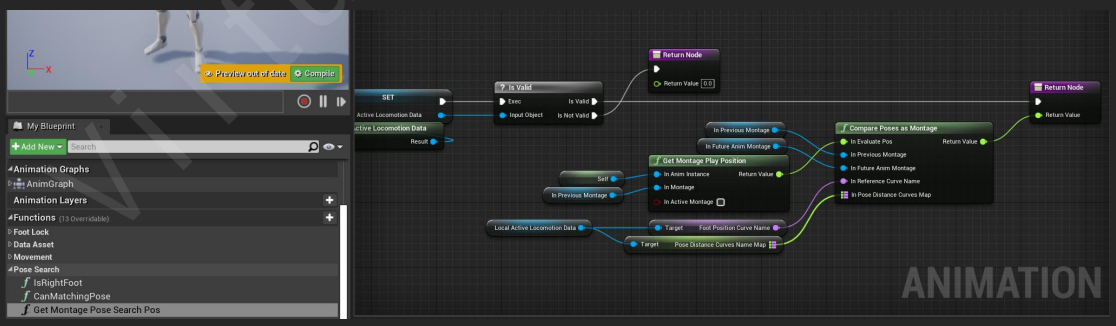
（4）脚步位置：

https://www.bilibili.com/video/BV1fa411i7Pq/?spm_id_from=333.788

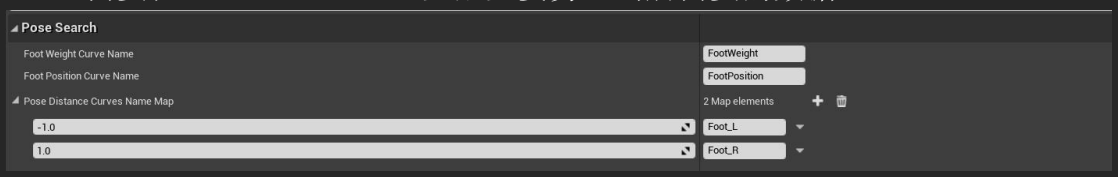
YouTuBe:
（1）脚步 IK：
https://www.youtube.com/watch?v=NbeXMBoWbf0&list=PLYP0ozRez418TgeOFVmyJSd87FIEBTPrg&index=30
（2）脚步锁定：
https://www.youtube.com/watch?v=ztq9zZu-6XA&list=PLYP0ozRez418TgeOFVmyJSd87FIEBTPrg&index=31
（3）脚步权重：
https://www.youtube.com/watch?v=67RhPq5cVUw&list=PLYP0ozRez418TgeOFVmyJSd87FIEBTPrg&index=32
（4）脚步位置：
https://www.youtube.com/watch?v=IIYhyh2mm5c&list=PLYP0ozRez418TgeOFVmyJSd87FIEBTPrg&index=33

15. 姿势查询

哔哩哔哩：
https://www.bilibili.com/video/BV1rY4y1H7bx/?spm_id_from=333.788
YouTuBe:
https://www.youtube.com/watch?v=x5KjB79JKJA&list=PLYP0ozRez418TgeOFVmyJSd87FIEBTPrg&index=28
（1）.IsRightFoot()：判断当前姿势的脚步范围，返回左脚或右脚（参考 VAT 脚步位置工具）
（2）CanMatchingPose()：判断是否匹配姿势（参考 VAT 脚步权重工具）
（3）Get Montage Pose Search Pos：获取当前蒙太奇匹配下一个蒙太奇的跳转位置

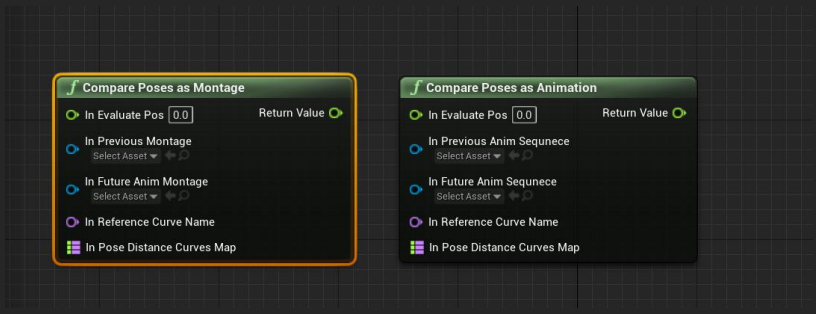


（4）需要在 Locomotion Data 里配置姿势查询所需要的数据

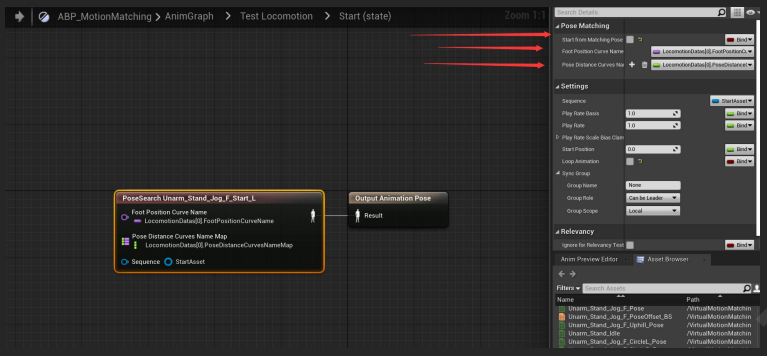


Virtual Game - Ryze

（5）可以使用这两个函数对两个动画资产进行姿势查询

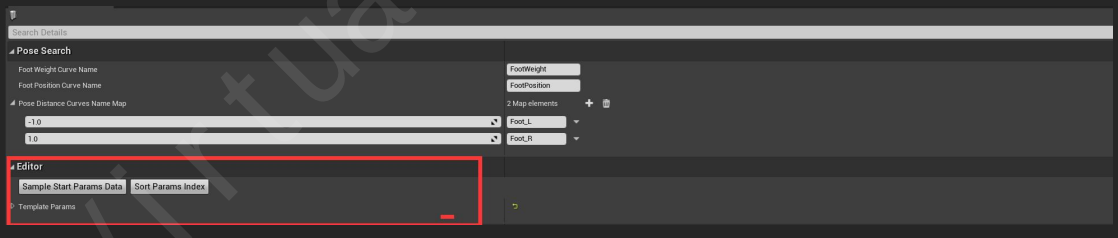


（6）可以在状态机里使用姿势查询(当前仅 4.27 使用)



16. 编辑器脚本

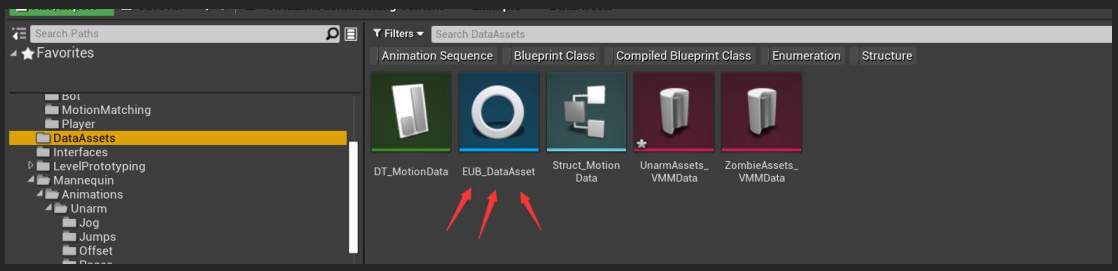
（1）可以在 C++自定义编辑器调用的脚本函数

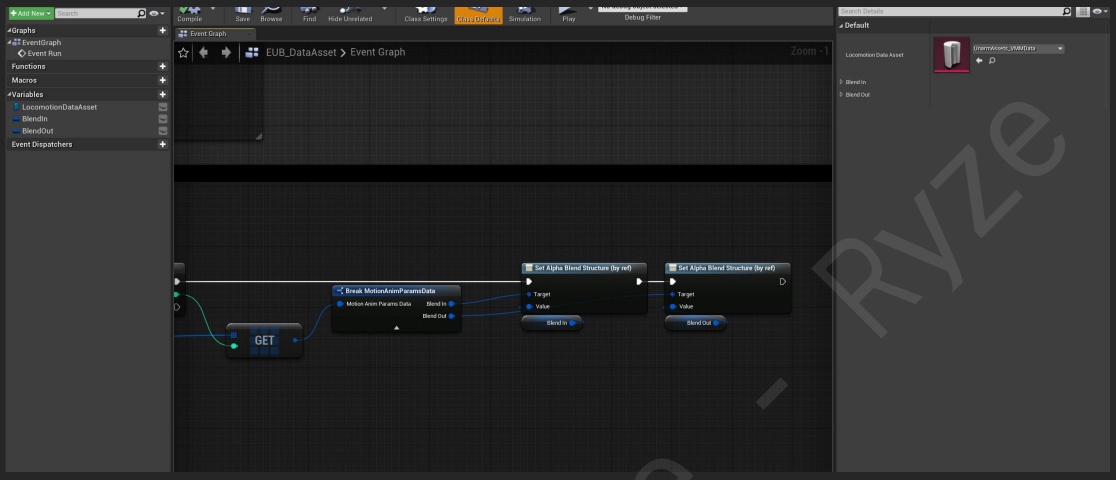



```
#if WITH_EDITOR
protected:

    /** An index that sorts all data to support fast queries. */
    UFUNCTION(CallInEditor, Category = Editor)
    void SortParamsIndex();

    /** Change start params data from template params. */
    UFUNCTION(CallInEditor, Category = Editor)
    void SampleStartParamsData();

#endif // WITH_EDITOR
```

Virtual Game - Ryze

（2）使用蓝图脚本批量处理动画数据（批量设置混入、混出等动画数据）



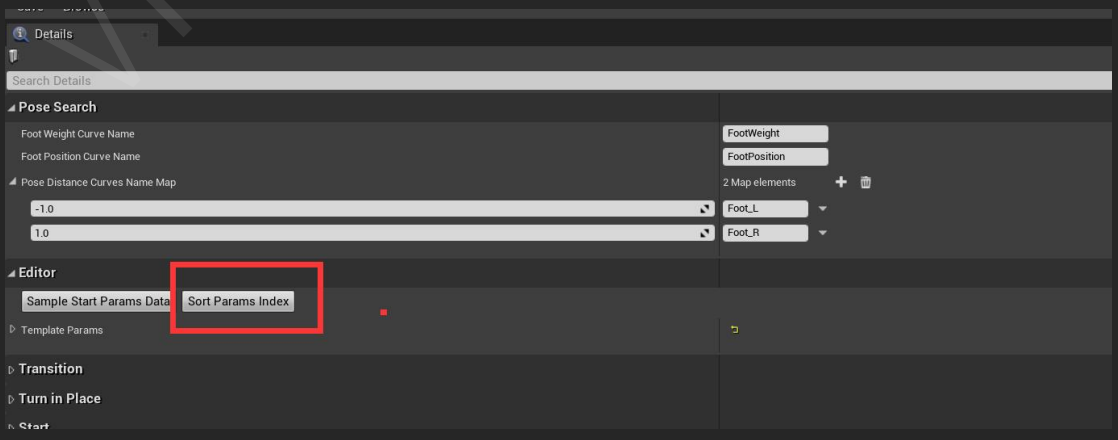


Virtual Game - Ryze

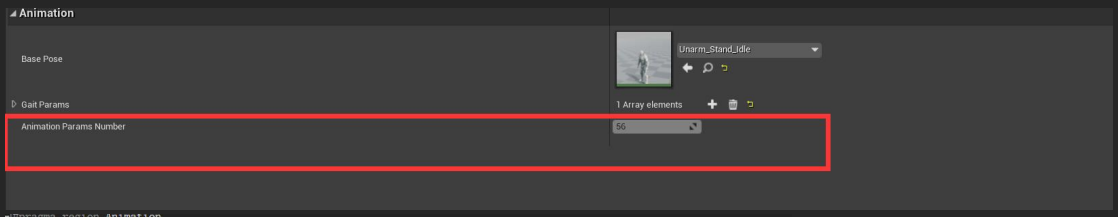
17. 动画数据及网络优化

（1）之所以使用数组而非 TMap 等数据结构是为了编辑器可视化以及代码性能上的优化

（2）每次配置完数据在运行前都应该点击该按钮进行索引排序，这是为了网络发包上的优化，我们只需要发送对应的 DataIndex 和 ParamsIndex 即可快速查询对应的动画数据

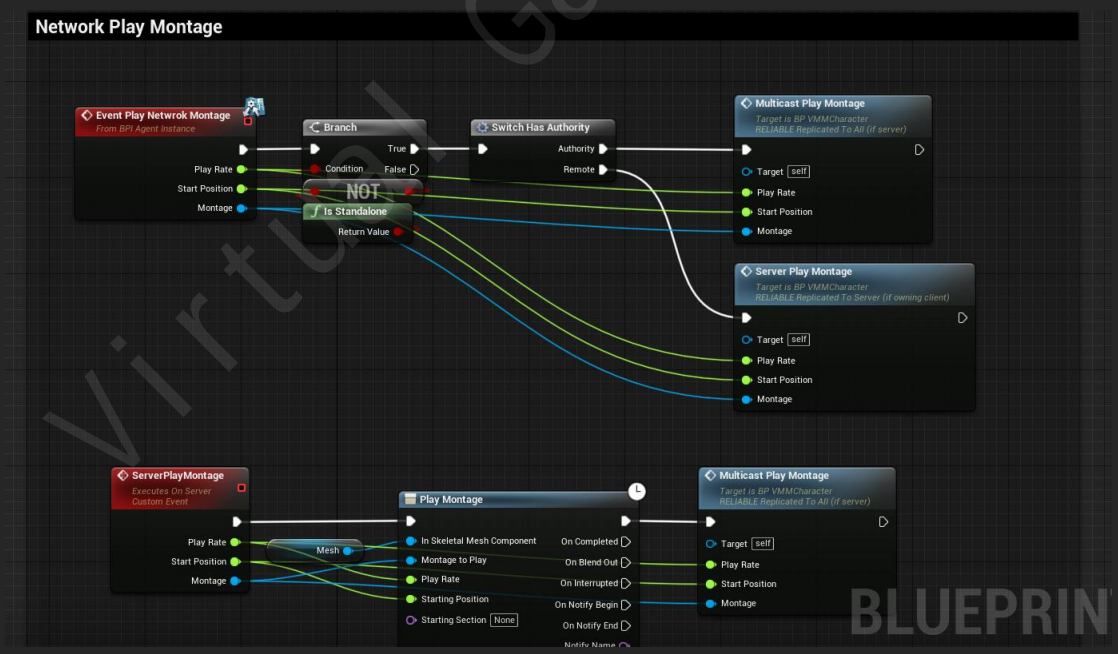


（3）为了防止编辑器数据过多导致卡顿等情况，这里只显示排序的动画数量，目前最大的数据量为 256 个，即 1 字节



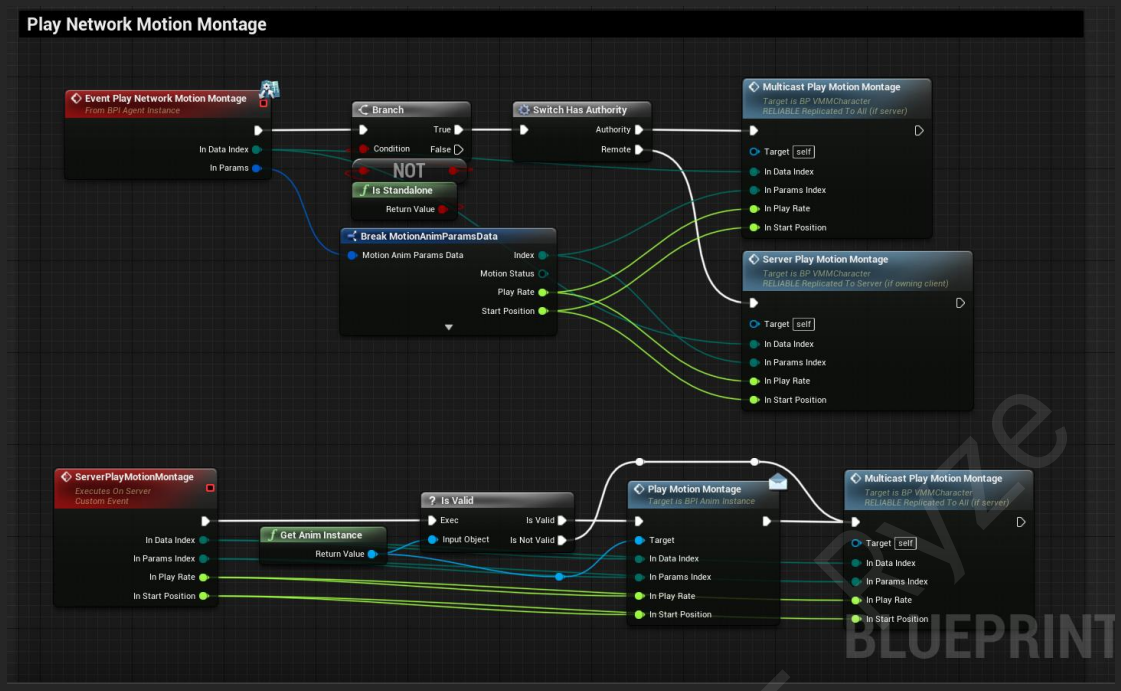

```cpp
#pragma region Animation

protected:

    /** Idle animations. */
    UPROPERTY(EditAnywhere, BlueprintReadWrite, Category = Animation)
    UAnimSequence* BasePose;

    /** Gait animation in place, we can customize the index, Start, Moving, Stop, Pivot, Jump etc.. */
    UPROPERTY(EditAnywhere, BlueprintReadWrite, Category = Animation)
    TArray<FMotionGaitAnimsData> GaitParams;

    /** Number of animations currently collected */
    UPROPERTY(VisibleDefaultsOnly, Category = Animation)
    int32 AnimationParamsNumber;

    /*
    * Collect all animation params so that you can quickly get the specified params according to the index
    * Not displayed in the editor, because it may cause lag
    */
    UPROPERTY(/*VisibleAnywhere, Category = Animation*/)
    TArray<FMotionAnimParamsData> MotionParams;

public:

    /** Return valid gait animation data index from input data */
    UFUNCTION(BlueprintPure, Category = Animation)
    const int32 GetValidGaitAnimsDataIndex(int32 InGaitAxis, const EMotionCardinalDirection::Type& InCardinalDirection);

    /** Return gait animation data from gait axis */
    FORCEINLINE const FMotionGaitAnimsData* GetGaitAnimsData(const int32& InGaitAxis)
    {
        return GaitParams.IsValidIndex(InGaitAxis) ? &GaitParams[InGaitAxis] : nullptr;
    }

    /** Return the motion params from animations array. */
```

（4）使用蒙太奇对象的发包，首次发包会有额外的数据量（动态创建的蒙太奇不应该使用该方式，或者需要在 C++将动态创建的蒙太奇设置为网络对象）



Virtual Game - Ryze

（5）使用 DataIndex 和 ParamsIndex 进行发包，可以大大减少发包的数据量



18. 运动调试及参数调试

（1）运动调试将在 V2 推出，具有运动匹配和姿势查询等的可视化调试
（2）参数调试,我们可以使用 TAB 键呼出对应的界面在游戏里实时修改对应参数（你可以定义自己喜欢的模板类）