VR IK Body Plugin

IKBodyData Struct

All transforms are in World Space or relative to Player Pawn origin if both **VRInputFromComponents** and **FollowPawnTransform** flags are false.

MEMBER	TYPE	DESCRIPTION
Pelvis	*Transform*	Pelvis Transform
Ribcage	*Transform*	Ribcage/Spine Transform
Neck	*Transform*	Neck Transform
Head	*Transform*	Head Transform
UpperarmRight*	*Transform*	Right Upperarm Transform (only if ComputeHandsIK is true)
ForearmRight*	*Transform*	Right Forearm Transform (only if ComputeHandsIK is true)
HandRight	*Transform*	Right Palm Transform
UpperarmLeft*	*Transform*	Left Forearm Transform (only if ComputeHandsIK is true)
ForearmLeft*	*Transform*	Left Forearm Transform (only if ComputeHandsIK is true)
HandLeft	*Transform*	Left Palm Transform
ElbowJointTargetRight	*Transform*	IK Joint Target for right hand in world space (if ComputeHandsIK is true)
ElbowJointTargetLeft	*Transform*	IK Joint Target for left hand in world space (if ComputeHandsIK is true)
ThighRight*	*Transform*	Right Thigh Transform (only if ComputeFeetIK is true)
CalfRight*	*Transform*	Right Calf Transform (only if ComputeLegsIK is true)
ThighLeft*	*Transform*	Left Thigh Transform (only if ComputeLegsIK is true)
CalfLeft*	*Transform*	Left Calf Transform (only if ComputeLegsIK is true)
FootRightCurrent	*Transform*	Right Feet IK instantaneous Transform (only if ComputeFeetIKTargets is true)
FootLeftCurrent	*Transform*	Left Feet IK instantaneous Transform (only if ComputeFeetIKTargets is true)
IsJumping	*bool*	Flag indicates if character is jumping
IsSitting*	*bool*	Flag indicates if character is sitting
Velocity	*Vector*	Current Player Vector Velocity. Vector Length is equal to scalar speed (meters per second).
GroundLevel	*float*	Current Ground Z Coordinate

* - not updated via networking

VR IK Body Component parameters

VARIABLE	TYPE	DESCRIPTION
Setup		
ComputeFeetIKTargets	*bool*	Set this flag to True if you need Feet Transform Predictions (**FootRightCurrent**, **FootLeftCurrent**)
ComputeLegsIK	*bool*	Set this flag to True if you need thigh and calf transforms (**ThighRight**, **CalfRight**, **ThighLeft**, **CalfLeft**). Only works if ComputeFeetIKTargets is true.
ComputeHandsIK	*bool*	Set this flag to True if you need upperarm and forearm transforms and joint targets
		(**UpperarmRight**, **ForearmRight**, **ElbowJointTargetRight**, **UpperarmLeft**, **ForearmLeft**, **ElbowJointTargetLeft**).
UseActorLocationAsFloor	*bool*	If true, Owning Pawn location Z will be used as floor coordinate. This approach doesn't work properly on slopes and staircases. If false, uses Line Trace to find ground level.
FloorCollisionObjectType	*Collision Channel*	Collision object type of floor if UseActorLocationAsFloor is false.
VRInputOption	*VR Input Setup*	There are three options: Direct input from VR API, Input from scene components (camera and motion controller components), input from external variables updated in tick. If use components input or networking, call Initialize(...) function on begin play.
LockShouldersRotation	*bool*	If true, shoulders don't rotate to follow motion controllers location.
FollowPawnTransform	*bool*	This flag only works if VRInputFromComponents is false. If true, component uses Pawn Actor Transform to locate body in world space. Otherwise it retuns body relative to Pawn Origin. World space calculation is required for unnatural locomotion.
LeftPalmOffset	*Vector*	Palm location relative to Motion Controller Component transform (default value is for HTC Vive Motion Controller)
RightPalmOffset		
Body Params		

BodyWidth	*float*	Y-axis (right-left) body size (modified by body calibration)
HeadHalfWidth	*float*	Approximate head radius
HeadHeight	*float*	Approximate head height
SpineLength	*float*	Approximate distance from pelvis to neck (modified by body calibration)
HandLength	*float*	Approximate distance from collarbone to palm (modified by body calibration)
FootOffsetToGround	*float*	Distance from feet to ground, useful to correct skeletal mesh feet bones Z-offset
NeckToHeadsetOffset	*Vector*	Component Space Offset from Neck to Head (X is forward, Z is up), modified by body calibration
RibcageToNeckOffset	*Vector*	Component Space Offset from Ribcage to Neck (X is forward, Z is up)
MaxHeadRotation	*float*	Max permissible angle between head and pelvis Yaw rotations (used to correct pelvis rotation)
Networking		
ReplicateFullBodyState	*bool*	If true, the component calculates body state on local PC and send it to server for other connected users. If false, component sends Head and Hands transforms to server, and every client perform body calculation for each player locally. Choose first option (true) if CPU performance is a priority in your project. Choose remote calculations (false) to optimize a project for network bandwidth.
ReplicateInWorldSpace	*bool*	If ReplicateInWorldSpace is set to false (by default), all body data would be converted to actor space before replication and reconverted to world space on remote machines. This operation adds a lot of transforms calculations, but allow to use sliding (Onward-style) locomotion which implies that pawn locations on different PCs are slightly asynchronous at the same moment. Set ReplicateInWorldSpace to true if you don't use sliding player locomotion to optimize CPU usage.
SmoothingInterpolation Speed	*bool*	Speed of interpolation between current and received body state. Set to 0 to disable interpolation.

VR IK Body Component Functions

FUNCTION	RETURN VALUE	PARAMETERS
CalibrateBodyAtTPose CalibrateBodyAtIPose AutoCalibrateBodyAtPose	*bool*	(no params)
Calibrate Body Params at T-Pose (hand to the left and right) and I-Pose (hands down). Calibration will be applied automatically at the second call (order is not important). Returns true if calibration is finished successfully, i.e. after both calls. AutoCalibrateBodyAtPose() detects pose automatically.		

Initialize	*void*	**Camera**: Camera Component Reference **RightController**: Motion Controller Component Reference **LeftController**: Motion Controller Component Reference
Call this function on BeginPlay to setup component references if VRInputFromComponents is true		
ActivateInput	*void*	(no params)
This function activates VR IK Body component. If you use scene components for data input, use Initialize(…) function instead to initialize component references and activate component.		
DeactivateInput	*void*	(no params)
The function stops component. It can be reactivated later by Activateinput() call.		
ComputeFrame	*IKBodyData*	**DeltaTime**: float
Call in Tick(…) to calculate body state.		
GetLastFrameData	*IKBodyData*	(no params)
Returns a last body state struct calculated by ComputeFrame(...) . function		
ConvertDataToSkeletonFriendly	*IKBodyData*	**WorldSpaceIKBody**: IKBodyData
Converts calculated body parts orientation to skeleton bones orientation. It's useful to directly set bone transforms in Anim Blueprint using 'Modify (Transform) Bone' node. Keep in mind that transforms in the returned struct are still in world space. The function doesn't affects Pelvis. It always have X as forward axis and Z as up.		
ResetTorso	*void*	(no params)
Call this function to set Pitch and Roll of Pelvis to zero and Yaw equal to Head Yaw.		
GetCharacterHeight	*float*	(no params)
Returns calculated or calibrated character from feet to HMD. height		
GetCharacterLegsLength	*float*	(no params)
Returns calculated or calibrated legs length.		
AttachHandToComponent	*bool*	**Hand**: Controller Hand

		Component: Primitive Component **SocketName**: Name **RelativeTransform**: Transform
Function attaches hand palm to primitive component. Calculated as relative transform to component's socket (or relative transform to component itself if socket isn't specified). Affects Hand Transform. Affects Upperarm/Forearm Transforms and elbow joint target if ComputeHandsIK is true. Returns true if succeed.		
DetachHandFromComponent	*void*	**Hand**: Controller Hand
Reattach hand palm to motion controller.		
UpdateInputTransforms	*void*	**HeadTransform**: Transform **RightHandTransform**: Transform **LeftHandTransform**: Transform
Call this function in Tick() to update Head and hands transforms if VRInputOption is 'Input from Variables'		
GetCalibratedBody	*CalibratedBody*	(no params)
Returns the struct describing body calibration results. Use it to save and restore body params if you need to respawn player.		
RestoreCalibratedBody	*void*	**BodyState**: CalibratedBody
Loads body calibration parameters from CalibratedBody struct.		
ResetCalibrationStatus	*void*	(no params)
Mark body as non-calibrated. Function (replicated) keeps existing body params, but allow to recalibrate body if necessary.		
IsBodyCalibrated	*bool*	(no params)
Returns true if calibration is complete or calibration data is loaded by RestoreCalibratedBody		
IsValidCalibrationData	*bool*	**BodyParams**: CalibratedBody
Check if CalibratedBody variable is valid. Use it if you save and load calibration params.		

Events (this feature is not included in the retail version)

OnJumpStarted	(no params)	Event called when player starts a jump
OnGrounded	(no params)	Event called when player ends a jump
~~OnHeadShake~~	**Iteration**: float	Event called when player shakes a head
~~OnHeadNod~~	**Iteration**: float	Event called when player nodes a head
OnSitDown	(no params)	Event called when player finishing squatting
OnStandUp	(no params)	Event called when player stands up
OnCalibrationComplete	(no params)	Event called when player body calibration complete successfully

VR IK Skeletal Mesh Translator

Component extracts data from VR IK Body Component and returns PoseSnapshot object ready-touse with any custom skeletal mesh.

Parameters

VARIABLE	TYPE	DESCRIPTION
RootBone, Pelvis, Ribcage, Head, ShoulderRight, UpperarmRight, LowerarmLeft, PalmRight, ShoulderLeft, UpperarmLeft, LowerarmLeft, PalmLeft, ThighRight, CalfRight, FootRight, ThighLeft, CalfLeft, FootLeft	*Name*	Names of a bones in a skeleton object. RootBone must be equal to Pelvis (not None) if Pelvis is a root. Ribcage is the last spine bone.
RescaleMesh	*bool*	Should Component apply player's height to skeletal mesh scale or not. If true, scale applied on VRIKBody component's OnCalibrationComplete event.
RootMotion	*bool*	If true, moves both Root bone and Skeletal Mesh Component.

Functions

FUNCTION	RETURN VALUE	PARAMETERS
Initialize	*bool*	**ControlledSkeletalMesh**: Skeletal Mesh Component, **VRIKBodyComponent**: VR IK Body Component
Initialize references and build information about controlled skeletal mesh. References can be automatically extracted from the owner actor If it has a VRIKBody component and only one Skeletal Mesh Component. Skeletal Mesh Component must contain a Skeletal Mesh object in T Pose in a moment of initialization. Returns true if initialization was successful.		
IsInitialized	*bool*	(no params)
Is component initialized or not?		
GetSkeletalMeshSetup	*FSkeletalMeshSetup*	(no params)

Get current skeletal mesh bones setup. Use this function to save/restore mesh data without reinitialization.		
RestoreSkeletalMeshSetup	*SkeletalMeshSetup*	**SkeletalMeshSetup**: FSkeletalMeshSetup
Load skeletal mesh bones setup to component. Use this function to save/restore mesh data without reinitialization.		
GetLastPose	*bool, PoseSnapshot*	(no params)
Get current pose from VRIKBody component adjusted for controlled skeletal mesh. Returns PoseSnapshot object and true if the component is initialized and calibrated.		
GetMeshWorldTransform	*Vector, Rotator*	(no params)
Return current Location and Rotation of Skeletal Mesh Component in World Space if Root Motion is enabled or predicted Location and rotation if Root Motion is disabled.		
UpdateSkeleton	-	(no params)
Must be called in Tick(…) after VRIKBody::CalculateFrame and before any artificial locomotion to load calculated data and update skeletal mesh position (if Root Motion is enabled)		
ForceSetComponentAsCalibrated	-	(no params)
Function marks body as calibrated and allow to receive input from VRIKBody component. Note: function isn't replicated, call it on all clients manually.		

How to use

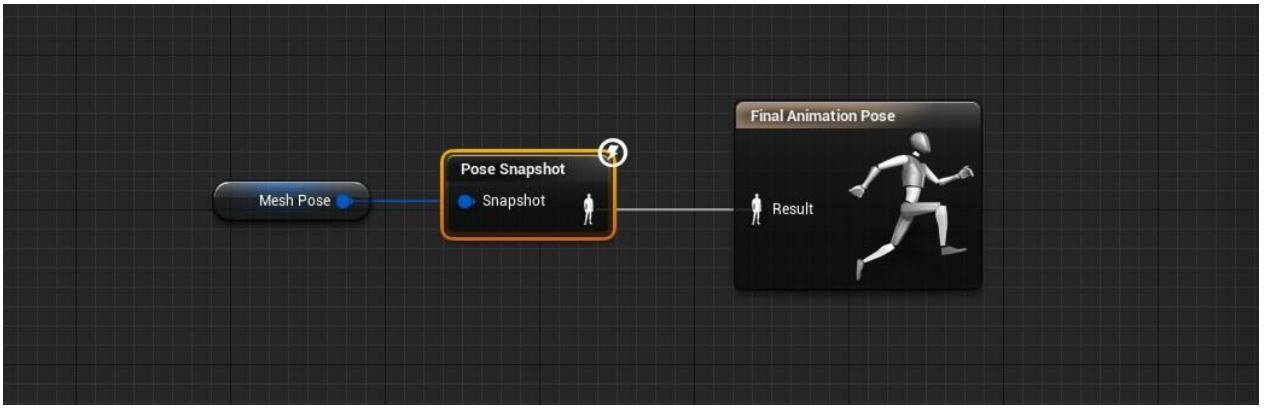
Add VR IK Body component to player pawn blueprint and setup required params. Call **Initialize(…)** function on BeginPlay if you want to use camera and motion controller components for input or **ActivateInput()** if you don’t want (and **VRInputFromComponents** is false).

Add VR IK Skeletal Mesh Translator component to player pawn and initialize it with references to VR IK Body component and skeletal mesh component. Keep in mind, that skeletal mesh must be in a reference pose (T-pose) when you call **Initialize(…)**.

On Tick Event call **ComputeFrame(…)** function. Body Plugin.

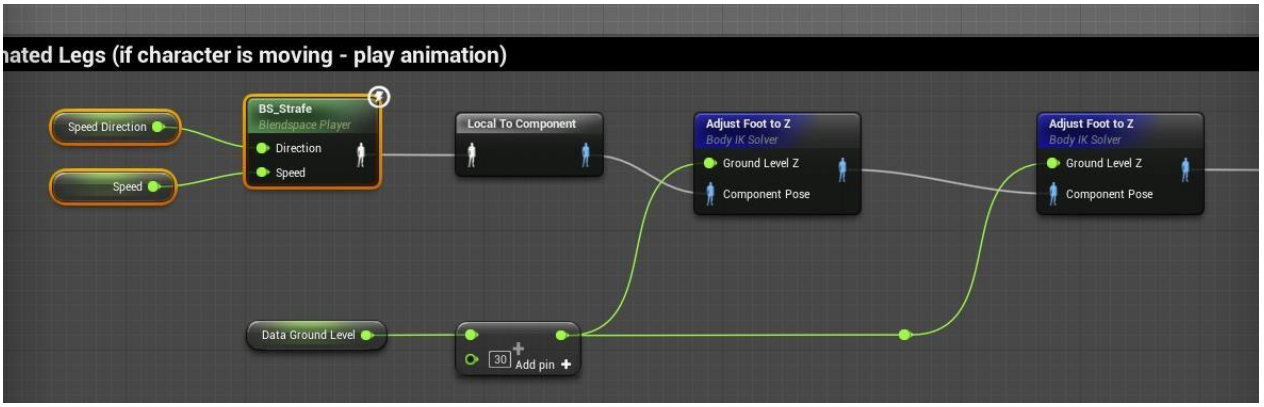
To calibrate body, ask player to take T-Pose and I-Pose alternately and execute some action (for example, press motion controller button) and call **CalibrateBodyAtTPose()** and **CalibrateBodyAtIPose()**.

In Animation Blueprint call GetLastPose() function of VR IK Skeletal Mesh Translator. You can use PoseSnapshot directly to animate skeletal mesh in Anim Graph, but default legs cycle is far from perfect.

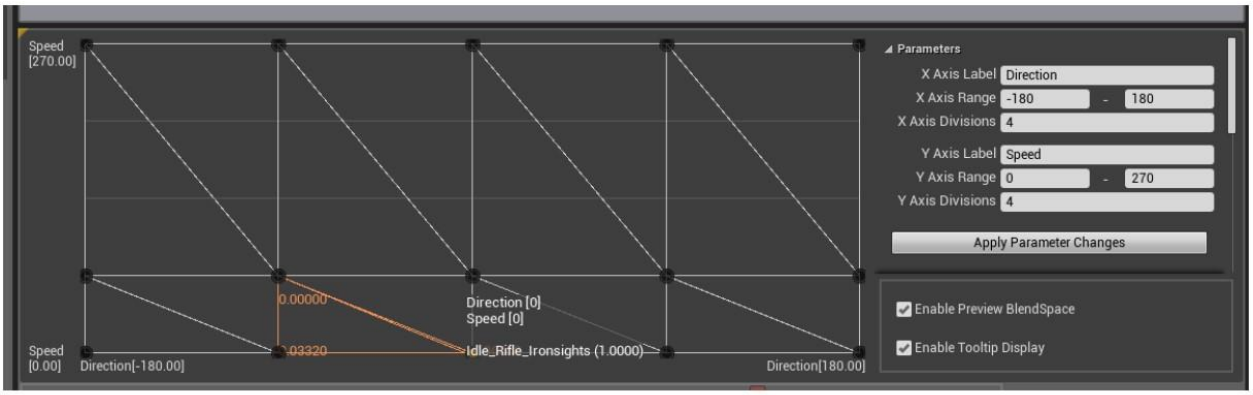


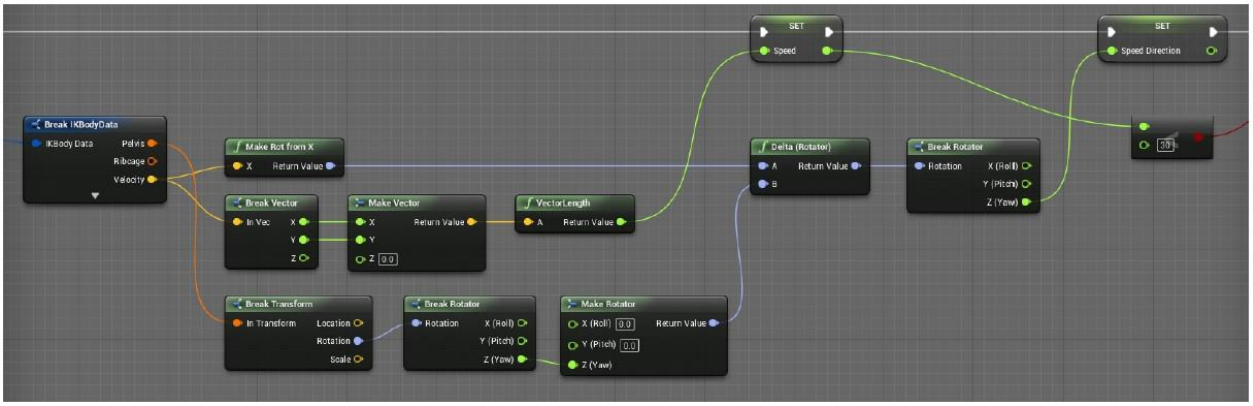
I recommend to use an animation assets (for example, strafe blend space) to get a better quality walking animation. To do that setup a strafe blend space and use **FIKBodyData::Velocity** vector to get direction angle and scalar speed. You have to adjust feet location after animation apply to make your skeletal mesh walk at the ground regardless of pelvis location above the ground. The plugin contains custom animation node **Adjust Foot To Z** for this purpose. This node is based on **Two Bone IK** node and uses Knee location as a joint target and current foot coordinates (obtained from the animation) with adjusted Z –axis value as effector location. When you get working lower body pose, blend it with upper body from the Pose Snapshot object.

Legs Animation:



Strafe Setup:





Notice: Plugin doesn't track roll and back spine leans.

Networking

1. Turn on replication in component Details panel.
2. In multiplayer, camera and controllers on remote PCs must be detached from HMD and motion controllers. To do that, you need to call Initialize(…) function on start. Use only Camera Component and Motion Controller components as a parameters and not meshes attached to this components. Of course, you also can to perform this setup manually.
3. To save and restore calibrated body, use GetCalibratedBody and RestoreCalibratedBody functions.

In other aspects networking setup is similar to single player. Keep in mind that the replication feature is in beta version.