



NABLA VR

VR IMMERSIVE HANDS v2.2

DOCUMENTATION

CONTENTS

OVERVIEW	6
ANIMATED HANDS AND CONTROLLERS (HAC)	7
1. BPAC_HAC/BPAC_HAC_GS	9
Spawn HACs/Spawn HAC_GSs	9
Update HAC	10
Show Controller Hint	12
Show Directional Hint	13
Show Hand Hint	13
Clear Hints	14
Emote	14
Update Laser	14
2. BP_HAC/ BP_HAC_GS	15
Soft Reference	15
Update Fingers	16
Update Interp Speed	17
Auto Scroll	17
3. BP_Hint	18
MPC_HintCollection	19
GRAB SYSTEM	20
1. BP_HAC_GS	22
2. BPAC_GS_Core	23

3. Grab Points	24
Creating Grab Anim with Control Rig	25
4. BPAC_GS_Placeable	27
5. BPAC_GS_PlaceableBase.....	28
6. BPAC_GS_Respawn	29
7. BPAC_GS_PhysicsSound.....	31
OTHER FEATURES	32
1. Settings	32
Auto Detect.....	33
Enable VR	33
2. BP_VRPawn/BP_VRPawn_GS	34
3. VR_Console	34
EXPANDING THE CONTENT.....	35
1. How to Add New Hand.....	35
Default UE4 Hand Skeletons	35
Finding Hand Offset	36
Custom Hand Skeletons (Advanced)	37
2. How to Add New Controller	38
Importing the Mesh (Advanced).....	38
Creating Animation Assets (Advanced)	40
Adding it to the Data Table (Advanced)	43
Hand Animations (Advanced).....	45

Update Button Functions (Advanced).....	46
Input Mappings	47
MIGRATING	48
1. Cleaning Up Before Migrating.....	48
2. Migrating Input Mappings	48
3. If You Only Want HAC (Optional)	49
4. How to Use Your Own Pawn (Optional).....	49

VERSION HISTORY

Version	Date	Description
1.0	11.09.2020	- Content published on Marketplace.
1.01	13.09.2020	- Fixed a bug where a pure function was crashing the packaged builds on Oculus Quest.
1.02	05.10.2020	- Fixed Valve Index controller position.
2.0	31.10.2020	- Moved logics to actor components - Multiple grab points - Reverse grab - Grab priority - Grab actions - Various improvements
2.1	21.11.2020	- Greatly simplified migration. - Minor fixes.
2.1.1	06.12.2020	- 4.26 is now supported - Added Hand Pose Generator assets - Minor fixes and improvements
2.2	21.01.2021	- Added Hints feature - Added First Step level - Improvements to the Grab System - VR Console improvements
2.2.1	Soon	- Minor GS bug fixes - More options for Hint

Please contact us at support@nablavr.com or join us at [Discord](#) if you have any questions, suggestions or bug reports.

OVERVIEW

VR Immersive Hands is an optimized and easy to use system that adds immersiveness to VR applications. With this system, you can add animated controllers, animated hands and a realistic grabbing to your VR applications.

Main Features

- 1- Animated Hands and Controllers
- 2- Hints
- 3- Grab System

Extra Features

- 1- VR Pawn – Setup and ready to use for most projects
- 2- UI Laser – Toggleable laser with 1 function
- 3- Emote – Hand emotes with 1 function
- 4- Auto Detect HMD – Target multiple HMDs. Can only detect Oculus HMDs out of the box
- 5- VR Console – Send text to console with specified color from any BP with 1 function
- 6- Auto Scroll – Smoothly scroll any widget with your laser
- 7- Editor Script – Speed up repetitive task
- 8- Hand Control Rig – Significantly speed up animation creation

Note: The last two features needs **Editor Scripting Utilities** and **Control Rig** plugins from Epic Games to work. However, it's not a requirement and the content will work without them.

ANIMATED HANDS AND CONTROLLERS (HAC)

HAC is a feature that has been built with optimization in mind from the start. It uses **Soft References** and **BP Interfaces** to be as optimized as possible. **So, it only loads assets that are relevant.** If your VR application supports multiple headsets, you don't want Valve Index assets (controller, anims etc.) to be loaded when the headset is Oculus Quest.

HAC can be used as just a hand, just a controller or both together.

- When used as just a hand, each finger can be blended independently and grabbing system can be used.
- When used as just a controller, all buttons are animated according to the button press.
- When used as both, each finger is animated according to button capacitive touch and press.

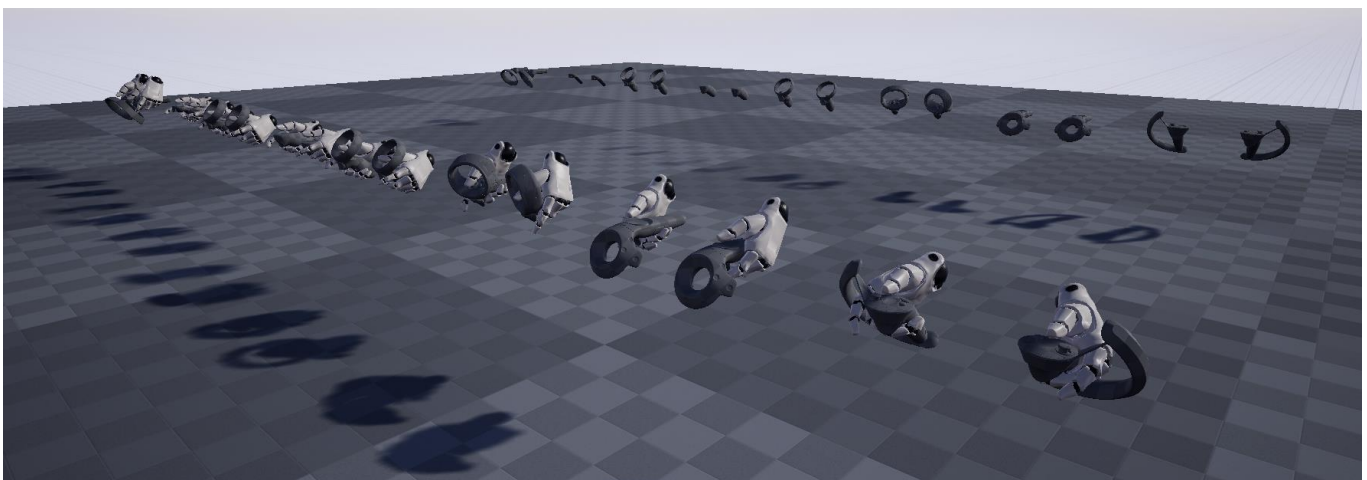
They can be set to be animated or not animated (optimized). Both hands are independent from each other. So, you can have different visibility on each one.





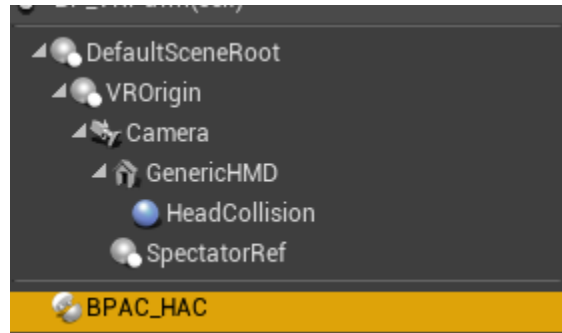
Click on the picture to go to Youtube video

Almost every controller on the market is supported out of the box. Here is a list;



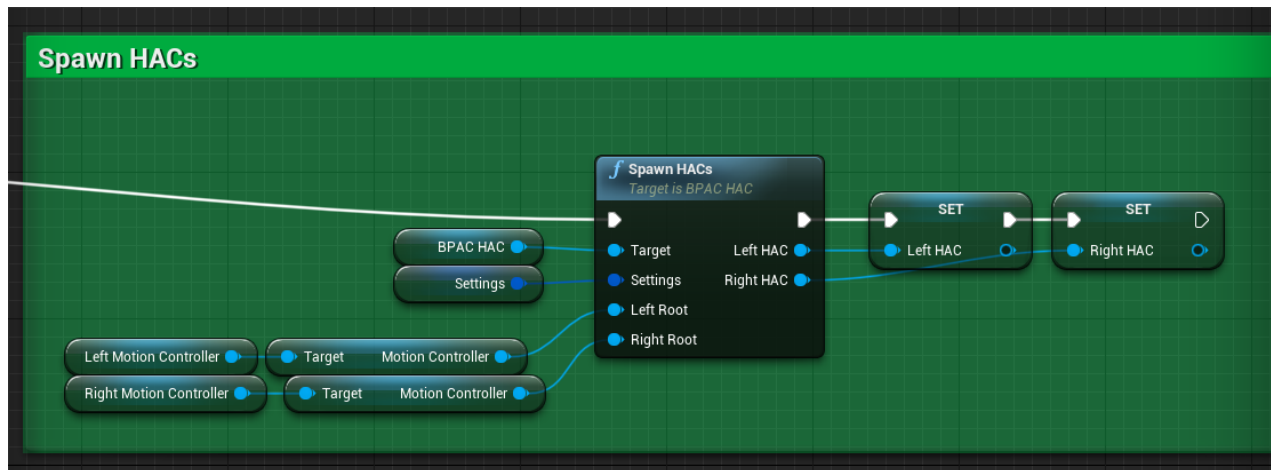
- Valve Index
- HTC Vive, Pro (**Tested**) *
- HTC Cosmos **
- Oculus Rift
- Oculus Rift S (**Tested**)
- Oculus Quest (**Tested**)
- Oculus Go ***
- Gear VR ***
- WMR

1. BPAC_HAC/BPAC_HAC_GS



These actor components contains everything needed to spawn and communicate with **BP_HAC** and its children. Just add this to your pawn and call **Spawn HACs** (Function) to get started. Use **BPAC_HAC** if you only want HAC, use **BPAC_HAC_GS** if you want the Grab System as well. **Don't** use them together.

Spawn HACs/Spawn HAC_GSs

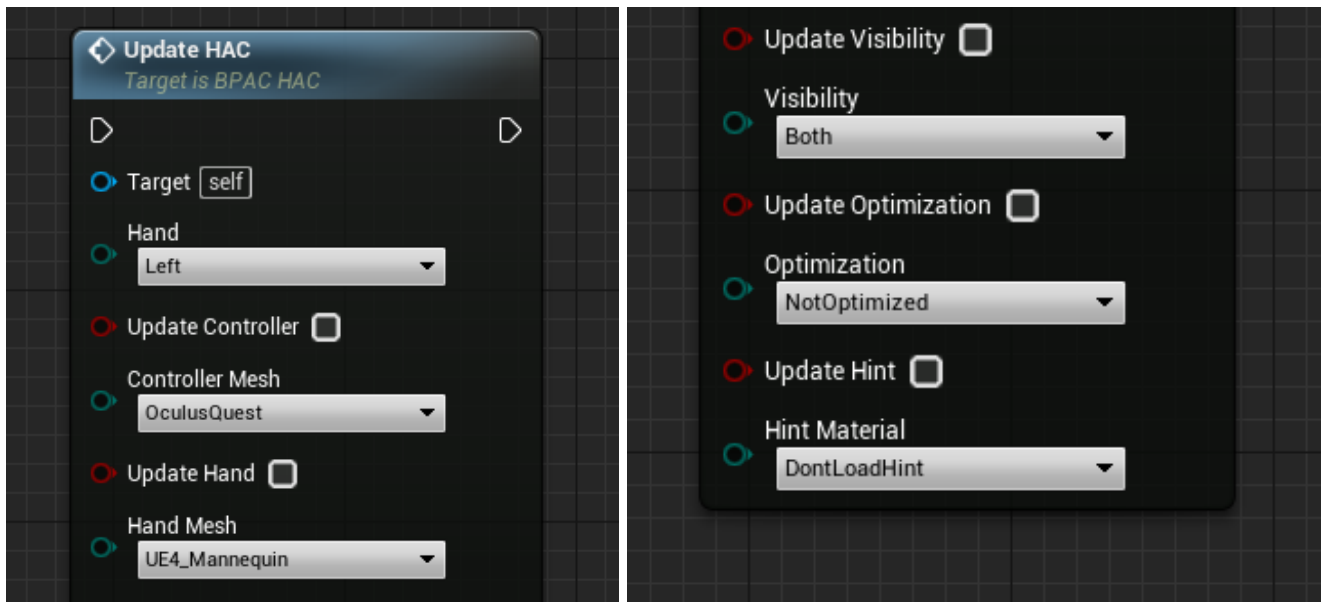


Input Variables

Settings	S_Settings	Settings to spawn HAC
Left Root	SceneComponent	Scene Component to attach Left HAC to. Generally Motion Controller component of Left MC
Right Root	SceneComponent	Scene Component to attach Right HAC to. Generally Motion Controller component of Right MC

This function spawns HACs and attaches them to the provided roots.

Update HAC



This function contains everything needed to initialize the **BP_HAC**. So you can use this function in runtime to change any settings in HAC. Don't forget that **Load Finished** (Event) will always be called after this function.

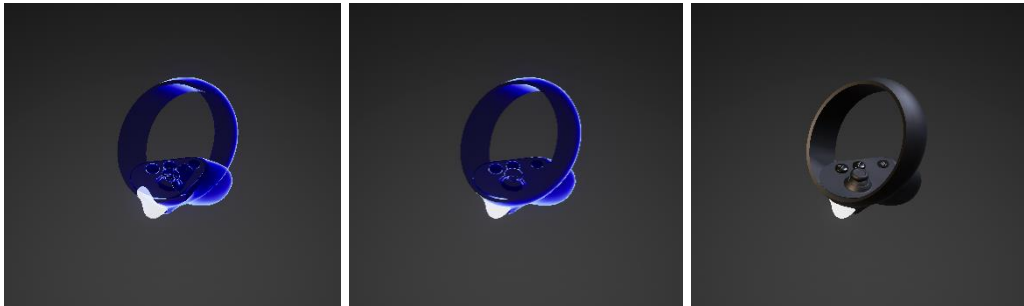


Visibility = Hands – Only the hands are visible in this mode. Each finger can be blended independently and have its own blend speed. Additionally, grabbing system and emotes only works in this mode.

Visibility = Controllers – Only the controllers are visible in this mode. Every button (including Thumbsticks) on the controllers are animated. An optimized (non animated static mesh) version can also be choosed.

Visibility = Both – Hands and controllers are both visible in this mode. All fingers are animated for buttons. Capacitive buttons are supported too. So when a player touches a capacitive button, corresponding finger will only touch the button, not press it.

Visibility = None – Nothing will be visible in this mode. This could be useful when loading a big level.



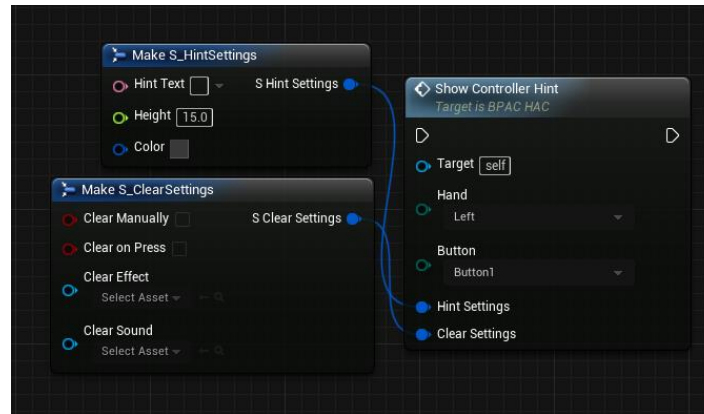
Hint Material = DontLoadHint – Nothing will be loaded in this mode.

Hint Material = Transparent – A transparent material will be loaded to use. Can only be used with Hands, Controller or None modes.

Hint Material = TransparentWithCulling – A transparent material with culling will be loaded to use. This material requires Custom Depth so it might not work on mobile platforms. If this is the case, this will be same as Transparent mode. Can only be used with Hands, Controller or None modes.

Hint Material = Highlight – A transparent material will be loaded to use. Can be used with every modes. This is the only one that can be used with Both mode.

Show Controller Hint

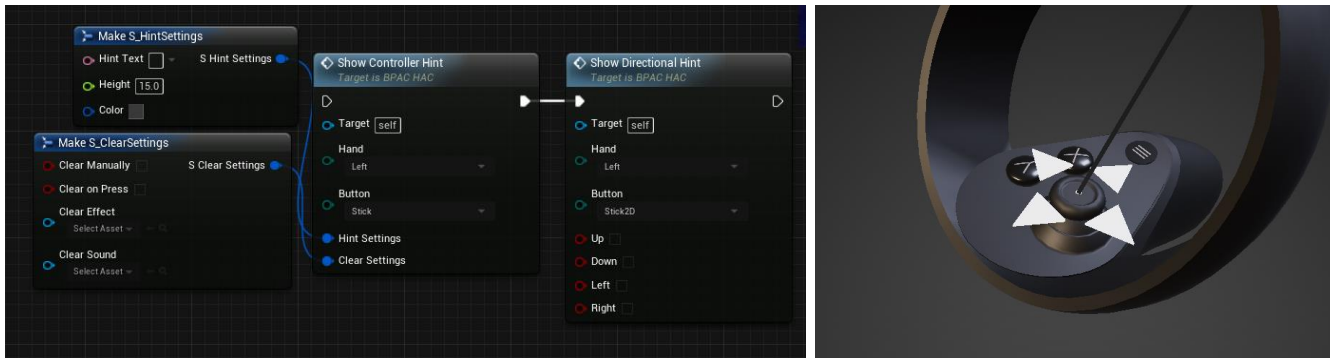


Hint Settings		
Hint Text	Text	Text to show. The text will scale down to fit to the widget if it's too long. Multiple rows are supported too.
Height	Float	Vertical distance between the button and the hint widget.
Color	Linear Color	Widget and line color.
Clear Settings		
Clear Manually	Boolean	Hints will be cleared automatically on button press or release if this is false. Otherwise, Clear Controller Hint or Clear All Controller Hints has to be called manually to clear.
Clear on Press	Boolean	Hint will be cleared on button Press if true, Release if false.
Clear Effect	Particle Effect	Effect to play on clear.
Clear Sound	Sound	Sound to play on clear. There are really good sound effects for this in the engine directory.

This function shows hints on your controller. If the controller doesn't have the specified button, it will warn. Controller hint **can be showed** even if the controller is not visible. If this is the case, controller will be hidden after all of the hints are cleared.

Stick Up, Down, Left, Right and Pad Up, Down, Left, Right **doesn't have any effect!** They are there because we didn't want to create a new enum just for this. Look at the **Show Directional Hint** to learn more.

Show Directional Hint



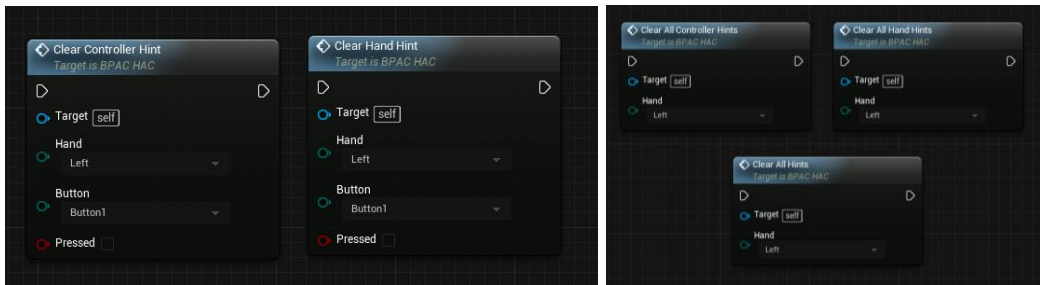
This function doesn't work on its own. There has to be a **Stick** or **Pad** hint already showed before this one is called. Otherwise it will warn.

Show Hand Hint



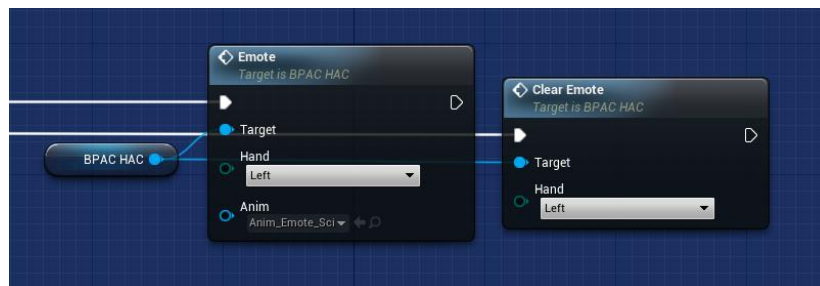
This function shows hints on your hands. If the controller doesn't have the specified finger, it will warn. Unlike controller hind, hand hint **can't be showed** if the hand is not visible.

Clear Hints



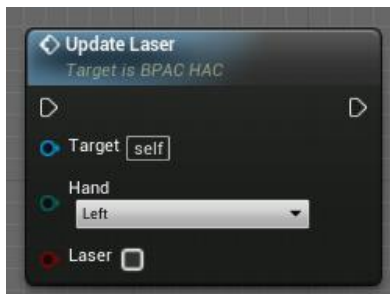
Functions to clear hints manually.

Emote



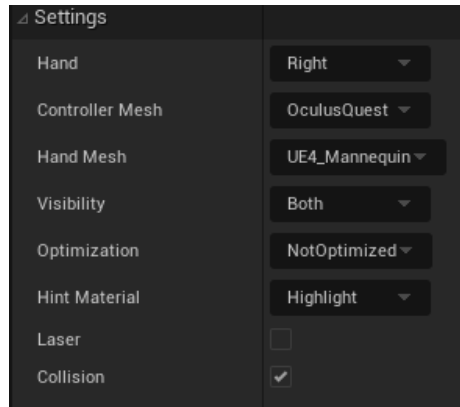
When **Visibility** is Hands, you can do hand emotes. You can find more example emotes in "HandsAndControllers/Mannequin/Anims/Emote"

Update Laser



Use this function to change which hand has laser to interact with UI and world.

2. BP_HAC/ BP_HAC_GS



Settings		
Hand	E_Hand	Sets which hand this BP should be
Controller Mesh	E_HMD	Sets which mesh should be used for displaying controller
Hand Mesh	E_HandMeshes	Sets which mesh should be used for displaying hand
Visibility	E_HACVisibility	Sets visibility of Hand and Controller
Optimized	E_Optimization	'Optimized' will load static mesh for controllers, 'NotOptimized' will load skeletal mesh for controllers
Hint Material	E_HintMaterial	Update HAC
Laser	Boolean	Sets whether laser should be active or not
Collision	Boolean	Sets whether collision should be enabled

The default UE4 hand mesh is too big for a normal human hand. That's why its scaled down to 0,75.

Setting **Optimization** to **Optimized** will load static mesh version of controllers for optimization.

Soft Reference

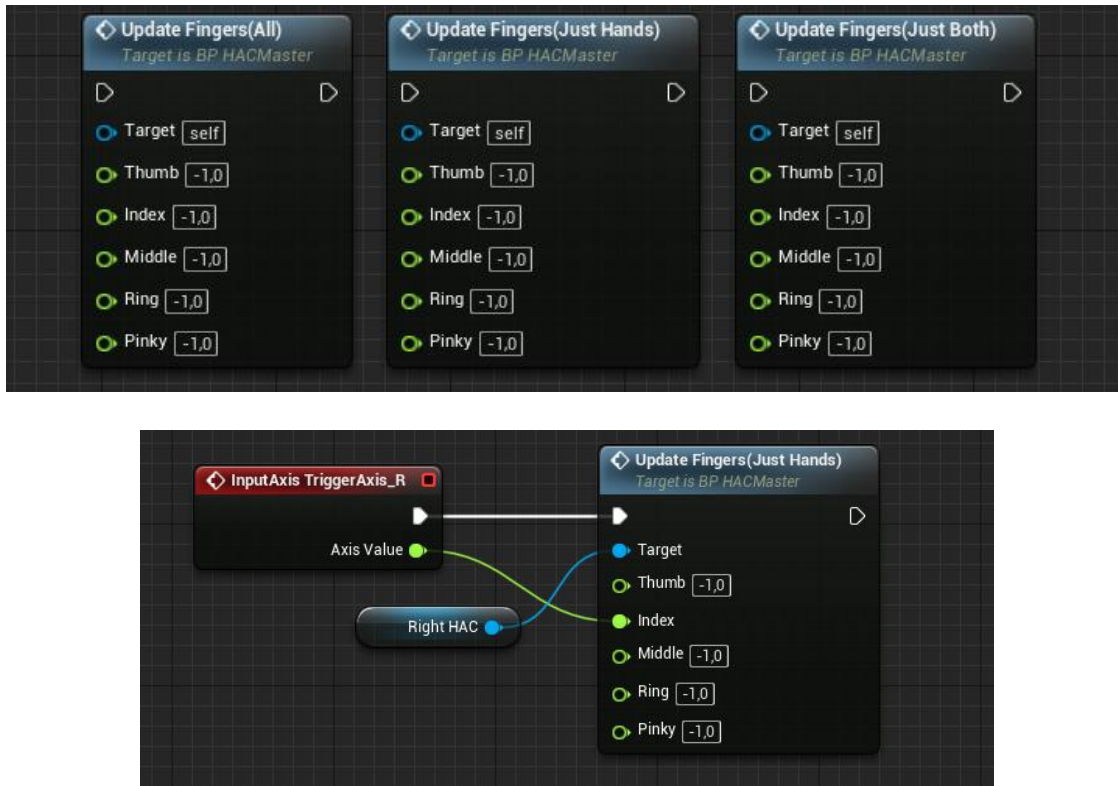
HAC uses Soft References to load only the necessary assets. Thus, we have to wait for assets to be loaded asynchronously to work on them.

On Begin Play, **BP_HAC** calls **Update HAC** (Function) to set visibility and start loading. Then, if there are no errors and everything is loaded, **Load Finished** (Event) will be called. You should treat the **Load**

Finished (Event) as the Begin Play for your logic. Otherwise, some of your logic might not reach their destination.

For example, if we try to set thumb position before loading is finished, Update Fingers Interface Message won't reach **AnimBP_MannequinHand** and it will **not** give you a warning.

Update Fingers

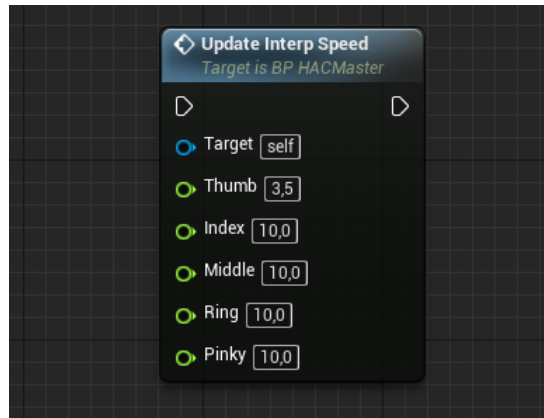


This function sets where each finger will be. 0 extends, 1 curls the fingers while -1 ignores input for that finger. All, Just Hands and Just Both are essentially the same thing. Difference is that Just Hands only works in Hands visibility, Just Both only works in Both visibility and All works in both. This will be handy if you want to use multiple visibility modes in the same application. Otherwise, you can use All.

Note: In Both visibility mode, 1 will rest the finger on the button for most cases except thumb. Only pressing a button will further move the finger.

For example, on Oculus Quest, Index value of 1 will **rest** the finger on the trigger, will **not** fully pull the trigger. Pressing the trigger button will further move the finger according to trigger axis.

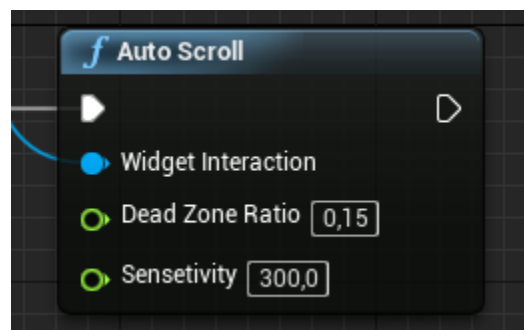
Update Interp Speed



This function sets blend speed for each finger. Animation blend will not work at 0, will be almost instant at 10 and will be ignored at -1. Default value is 3,5.

Note: You might want the Axis Event based animations more responsive and Boolean based animations to be slower. Index, Middle, Ring and Pinky are generally tied to Axis Events. While Thumb is tied to a Boolean Event.

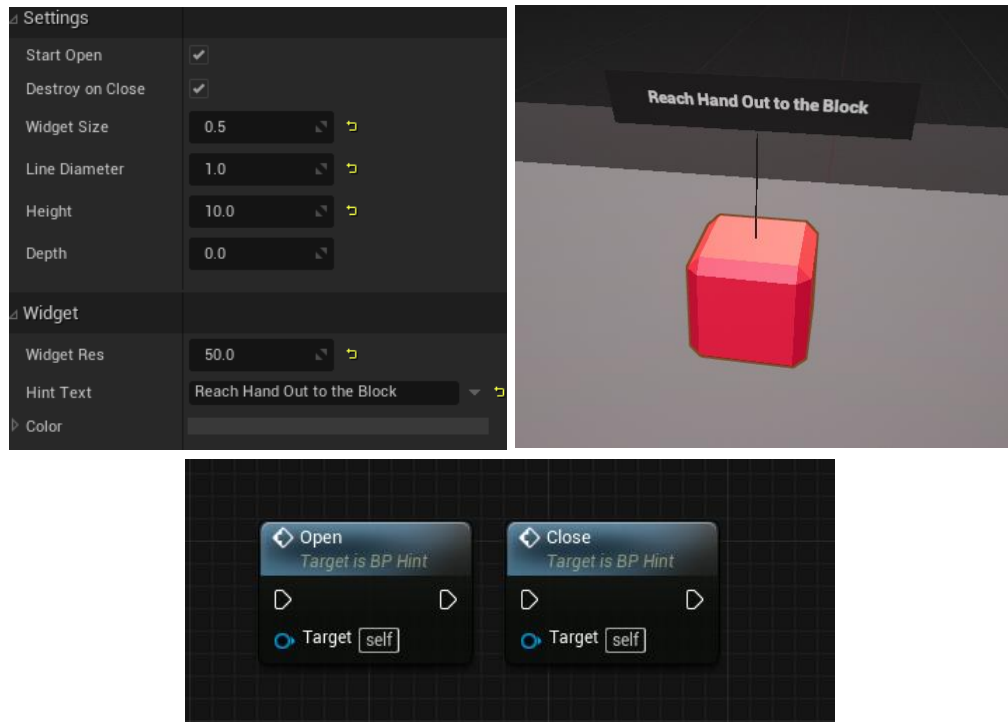
Auto Scroll



This feature allows you to smoothly scroll **VR Console** in runtime. For this to work, the lasers check every frame if they are pointed at a widget that implements **BPI_UI**. If they are, **Auto Scroll** (Function) will get its size and orientation then initiate scroll.

So, if you want auto scroll to work with your own widget, implement **BPI_UI** and set **Get Size** (Interface Function).

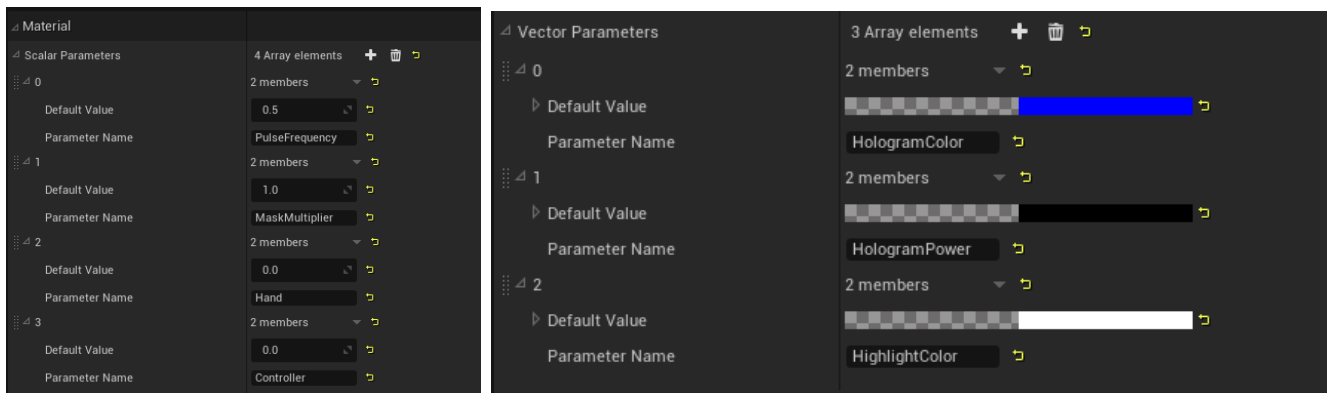
3. BP_Hint



Settings		
Start Open	Boolean	Sets whether hint should start open
Destroy On Close	Boolean	Sets whether actor should be destroyed on clear
Widget Size	Float	Size of the widget
Line Diameter	Float	Diameter of the line
Height	Float	Widget location
Depth	Float	Widget location
Widget		
Widget Res	Float	Widget resolution. Keep this value as low as possible while keeping the text readable. A bigger value will be more costly!
Hint Text	Text	Hint text
Color	Linear Color	Widget color

A hint actor to use in your level. The hints will always try to face the pawn. Don't use children actors of this BP manually. They are for controller and hand hint.

MPC_HintCollection



Float		
Pulse Frequency	0.5	Global pulse frequency
Mask Multiplier	1.0	0= No hint highlight, 1= Default, Bigger=More emmivise
Hand	0.0	1= Highlight whole hand, Bigger=More emmivise
Controller	0.0	1= Highlight whole controller, Bigger=More emmivise
Widget		
Hologram Color	Dark Blue	Hologram color towards outside
Hologram Power	Black	Hologram color towards inside
Highlight Color	White	Highlight color

This collection has every important variables for the hint feature. Have a look at the First Steps Level BP to see how it can be used.

GRAB SYSTEM

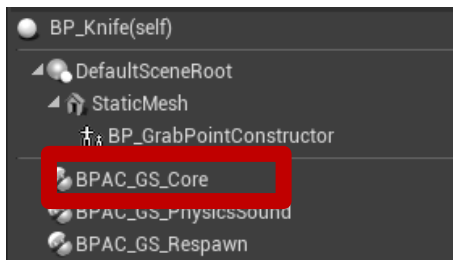
Grab System has been updated to a modularized component-based system to make integrating to existing projects much easier.

- Actors can be grabbed from multiple point.
- Every actor and their grab points has priority for grabbing.
- Actors can be held two-handed with reverse grabbing.
- Actors can be placed to compatible bases.
- Hands can extra play animations while grabbing an actor.
- Actors can respawn if they are below a certain Z value.
- Actors can have physics sound.

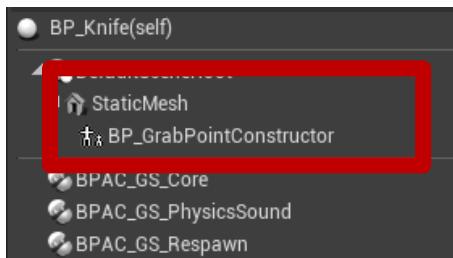


Click on the picture to go to Youtube video

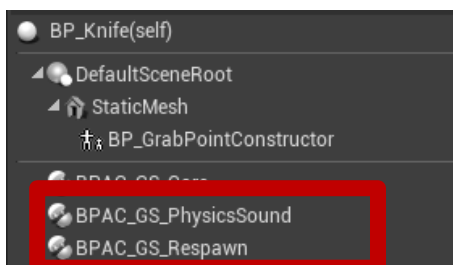
1. **REQUIRED** - Add **BPAC_GS_Core** component to any actor you want and set its variables.



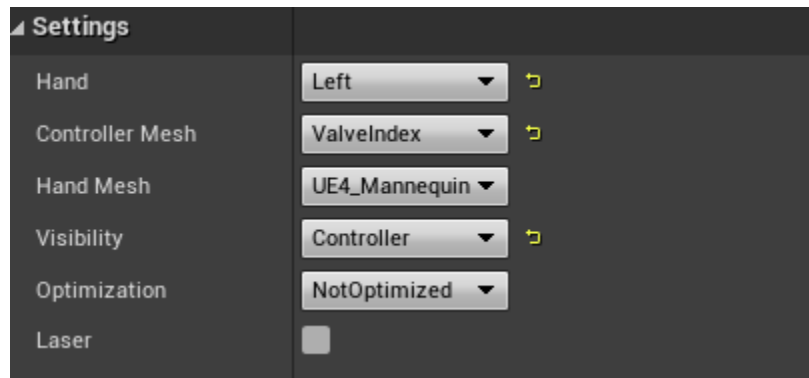
2. **OPTIONAL** - Add **BP_GrabPointConstructor** as child actor component to the actor and set its variables if you want to be able grab it.



3. **OPTIONAL** - Add any other component and set their variables.



1. BP_HAC_GS



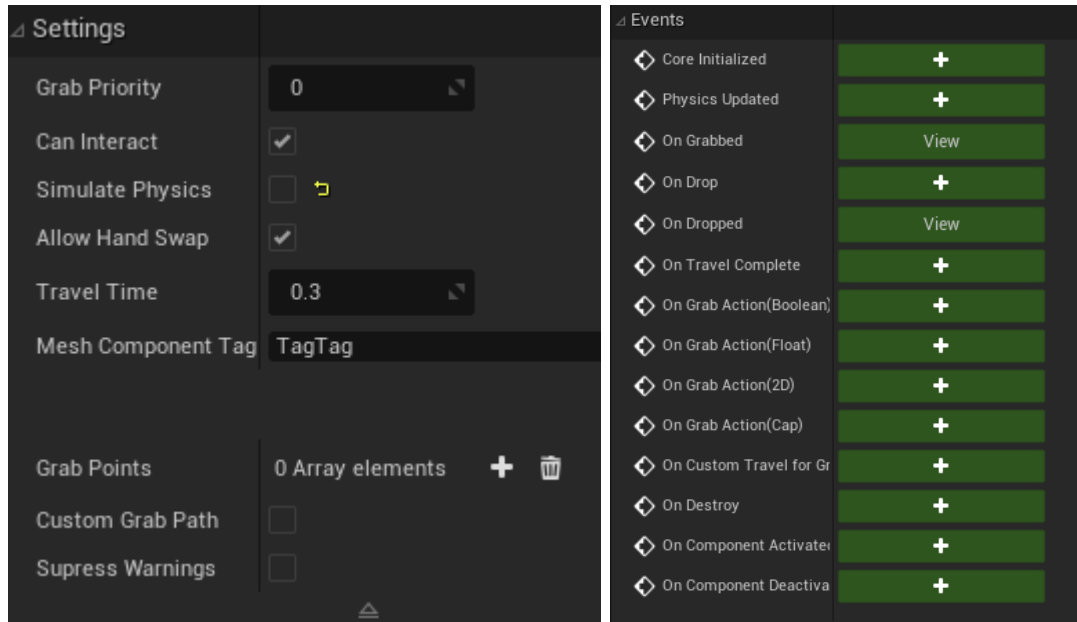
Settings

No new variable. Same as **BP_HAC**.

This is a relatively simple BP. It is a child of **BP_HAC** and expands on it to make grab system work.

Load Finished (Event) is overridden to make sure grabbing is reset on **HAC Update**. Delete the override if you want it the other way.

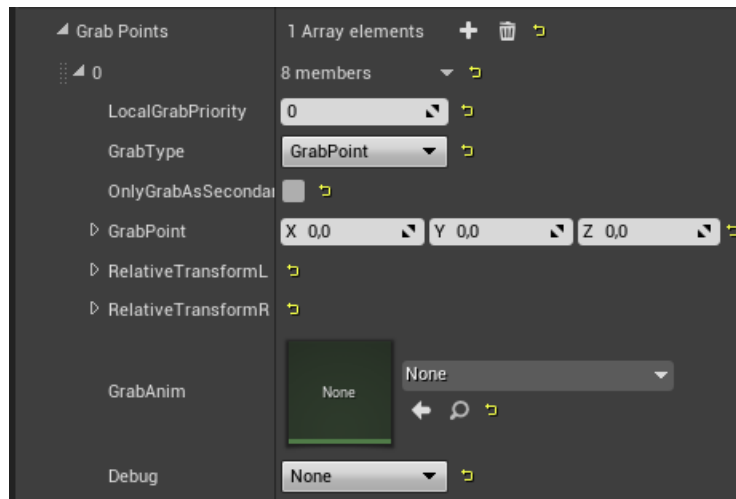
2. BPAC_GS_Core



Settings		
Mesh Comp Tag	Tag	REQUIRED – Add a tag to the Mesh Component you want and copy it here
Grab Priority	Integer	Actor level priority
Can Interact	Boolean	'true' if actor can be picked up, 'false' otherwise
Simulate Physics	Boolean	'true' if actor should simulate physics, 'false' otherwise
Allow Hand Swap	Boolean	'true' if actor should be allowed to swap hands, 'false' otherwise
Travel Time	Float	The time it takes the actor to lerp to the hand
Suppress Warnings	Boolean	'true' if actor should not send warnings or errors to output log, 'false' if otherwise
Grab Points	Array of S_GrabPoint	Grab Points
Custom Grab Path	Boolean	'true' if actor should follow custom path on grab, 'false' otherwise. OnCustomTravelForGrab can be used to start custom travel. Complete Travel has to be called after travel is finished.

This BP contains all the logic for a grappable object. But since actor components can't add component to their parent, **BP_GrabPointConstructor** has to be added to be able to grab an actor. It adds necessary components for grab system to its parent on begin play then destroys itself.

3. Grab Points



Grab Points		
Local Grab Pri.	Integer	Grab point priority
Grab Type	E_GrabType	Grab type
Only Grab as Sec.	Boolean	This will only be used with ReverseGrabPoint. If this is true, this point will only be grabbed if parent object is being held with other hand.
Grab Point	Vector	The location of the grab point
Relative Trans. L.	Transform	Creating Grab Anim with Control Rig
Relative Trans. R.	Transform	
Grab Anim	Anim Seq.	
Debug	E_GrabbingDebug	Hand preview for this point

There are 2 ways to setup Grab Points data of an actor. Both has its advantages and disadvantages due to engine limitations.

- 1- **With BP_GrabPointsConstructor:** ChildActorComponent variables can only be edited from its parents BP editor. This is the only way to preview hand grabs. Don't forget to click compile to update preview.
- 2- **With BPAC_GS_Core:** Actor component variables can be edited from everywhere. This means that this is the only way to set Grab Points directly from the level. Be aware, Grab Points variable in this component will override the one in the constructor. Since actor components can't add component to their parent, hand preview will not work in this way.

Creating Grab Anim with Control Rig

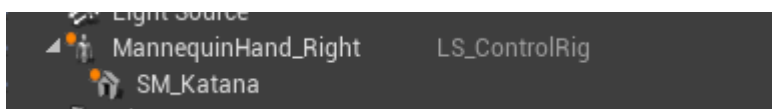


Click on the picture to go to Youtube video

Every grab point will probably need their own **Grab Animation** and **Relative Transform** to lerp it smoothly. We included an editor script and a Control Rig for finding the transform and speeding up animation creation.

Right click and select **Scripted Actions > Start Interactable Object Setup** on any static mesh that you want to use. This will;

- 1- Open **Map_Setup** map
- 2- Spawn the selected Static Mesh into level
- 3- Parent SM to Hand



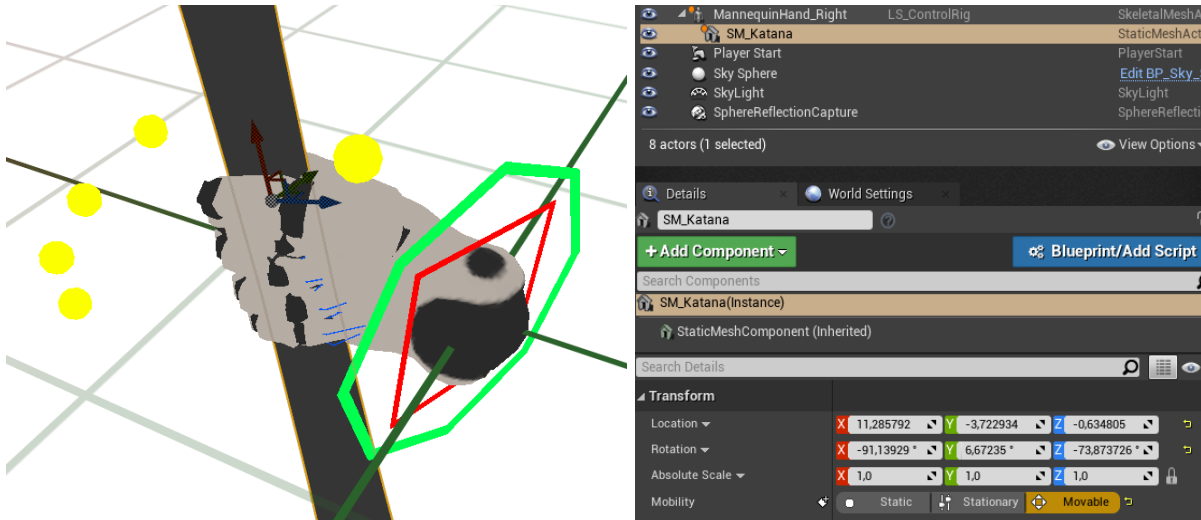
- 4- Open **LS_ControlRig**

Warning: Previous works in the map will be lost if you run this script! Open **Map_Setup** manually to return to previous work.

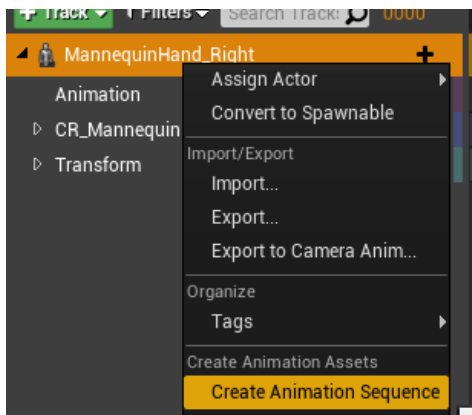
Warning: If you want to use Control Rig, 4.25+ is required to export animations.

Follow these steps to create animation and find the transform.

- 1- Find the most natural transform for hand to hold the object and place the fingers. Key frequently in LS so you don't lose progress. Use this transform as **Relative Transform R**. Ignore object's scale, which will be 1,333 (or 1,0 if its Absolute Scale).



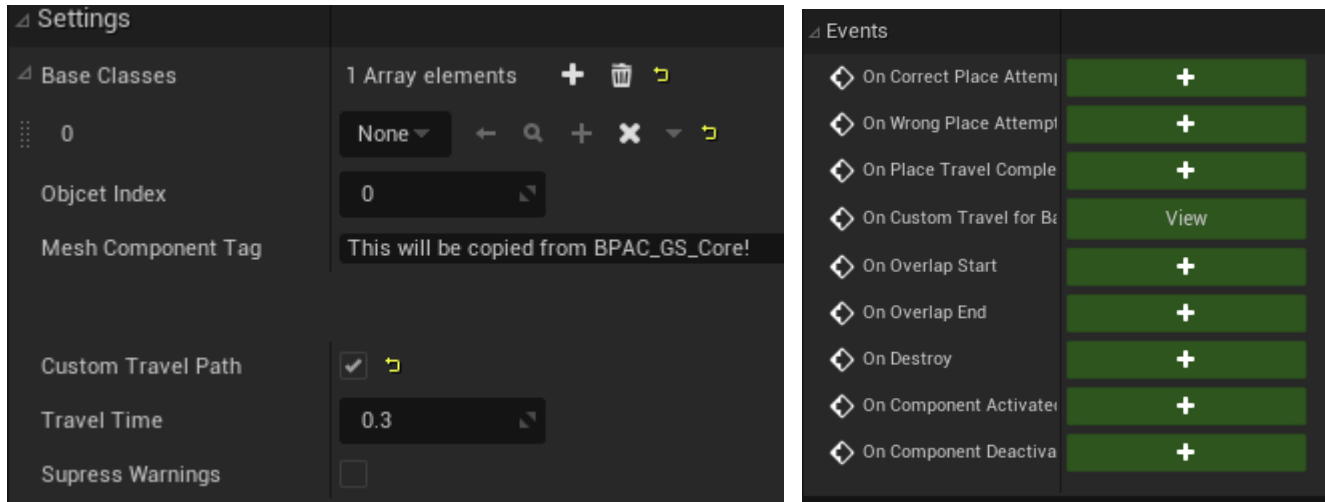
- 2- Place the fingers properly and save the animation by right clicking on MannequinHand_Right and clicking Create Anim Seq. **Animations should be 1 frame and the hand's transform must be 0,0,0 0,0,0 1,1,1.**



Note: Some objects will need different transform for left hand. Use **Relative Transform L** for this.

Warning: Sometimes the pivot of a control point will jump to 0,0,0 and moving it will move the hand as well. This is an engine bug. **Workaround:** Reselect the control point from **Animation** tab.

4. BPAC_GS_Placeable



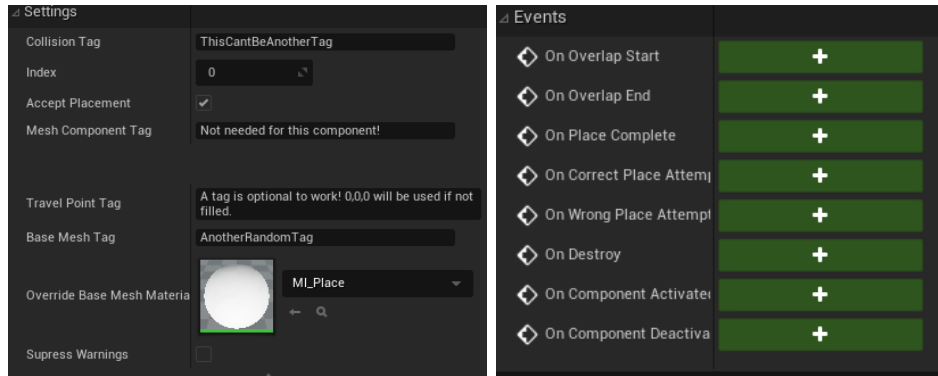
Settings		
Mesh Comp Tag	Tag	IGNORE for this component
Base Classes	Array of Actor Class	Which actor should this actor interact with
Object Index	Integer	Actors can only be placed if index is same with base's index
Custom Grab Path	Boolean	'true' if actor should follow custom path on placement, 'false' otherwise. OnCustomTravelForBase can be used to start custom travel. Complete Travel has to be called after travel is finished.
Travel Time	Float	The time it takes the actor to lerp to the hand
Suppress Warnings	Boolean	'true' if actor should not send warnings or errors to output log, 'false' if otherwise

With this component, actors can be placed on actors that have base component.



Meshes has to have **MF_Effect** (Material Function) in Emissive Color of all of its material for **Place Effect** (Event) to work as intended. Since it's optional, it won't give any warning if it's not implemented.

5. BPAC_GS_PlaceableBase



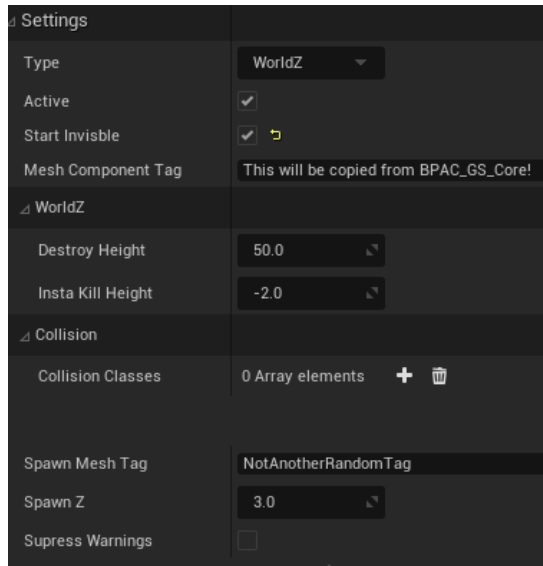
Settings		
Mesh Comp Tag	Tag	IGNORE for this component
Collision Tag	Tag	REQUIRED - Add a tag to the Collision you want and copy it here
Index	Integer	Base index
Accept Placement	Boolean	Accept Placement
Travel Point Tag	Tag	OPTIONAL - Add a tag to a scene component to lerp placeables to. This can be dynamic and move during placement. 0,0,0 will be used if its empty.
Base Mesh Tag	Tag	OPTIONAL - Add a tag to the Base Mesh you want and copy it here. This will be used for hover and placing animations.
Override Base M.	Material I.	Leave empty if you don't want to override
Supress Warnings	Boolean	'true' if actor should not send warnings or errors to output log, 'false' if otherwise

With this component, actors that have placeable component can be placed on it.



If **Override Base Mesh Material** is false, the mesh has to have **MF_Efect** (Material Function) in Emmisive Color of all of its material for **Place Effect** (Event) to work as intended. Since it's optional, it won't give any warning if it's not implemented.

6. BPAC_GS_Respawn

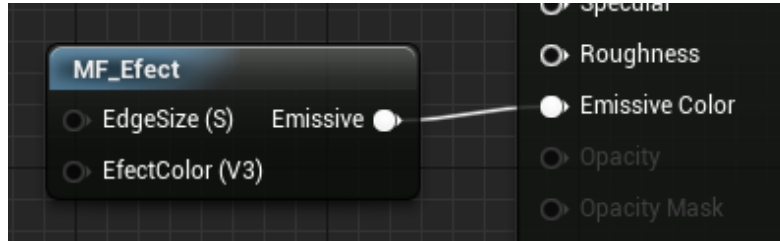



Settings		
Mesh Comp Tag	Tag	IGNORE for this component
Type	E_RespawnType	Type of respawn check to use
Active	Boolean	Whether the component is active or not
Start Invisible	Boolean	'true' if actor should be invisible at begin play, 'false' if not
Spawn Mesh Tag	Tag	OPTIONAL - Add a tag to the Spawn Mesh you want and copy it here. This will be used for respawn animations.
Spawn Z	Float	Spawn Mesh travel amount
Suppress Warnings	Boolean	'true' if actor should not send warnings or errors to output log, 'false' if otherwise
WorldZ		
Destroy Height	Float	Z value to start respawn countdown
Insta Kill Height	Float	Z value to respawn actor immediately
Collision		
Collision Classes	Array of Actor	Any actor you want to use as collision

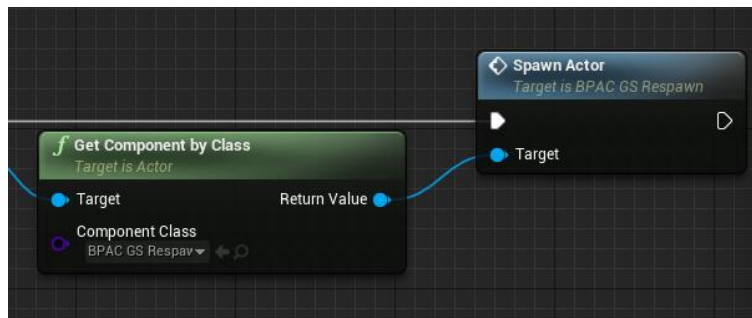
Event Tick will be enabled if **Active** is 'true' and disabled if its 'false'. It will constantly check the actor velocity and its World Z is against **Destroy Height** and **Insta Kill Height**.

If actor World Z is less than **Insta Kill Height**, the actor will be destroyed (Fake Destroy) immediately.

If actor World Z is less than **Destroy Height** but greater than **Insta Kill Height**, logic will wait until it settles down (velocity=0) then the actor will be destroyed (Fake Destroy).



If **Spawn Mesh Tag** is filled, the mesh has to have **MF_Effect** (Material Function) in Emmissive Color of all of its material for **Spawn Effect** (Event) and **Destroy Effect** (Event) to work as intended. Since it's optional, it won't give any warning if it's not implemented.

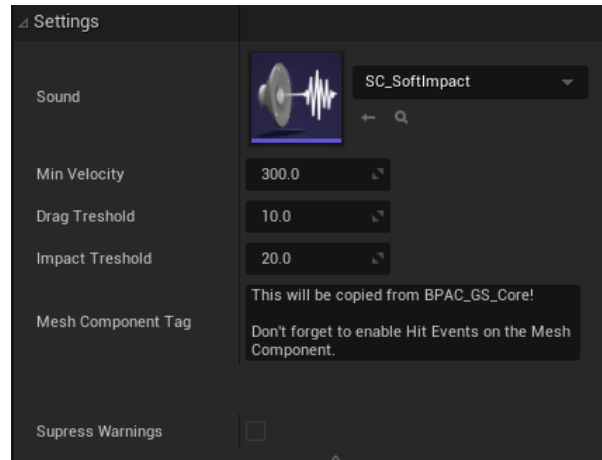


If **Start Invisible** is true, the object will start hidden and will not be interactable. To make the object visible and interactable, **Spawn Actor** (Event) has to be called.



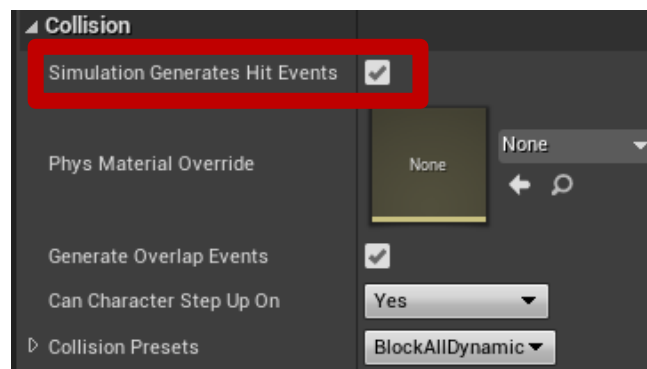
Spawn Mesh is an outline of the object mesh on Z axis. You should use **MI_Spawn** or a child of it as material for effects. Since it's optional, it won't give any warning if it's not implemented.

7. BPAC_GS_PhysicsSound



Settings		
Mesh Comp Tag	Tag	IGNORE for this component
Sound	Sound Base	Sound to play
Min Velocity	Float	Min velocity to consider for sound
Drag Threshold	Float	Sound won't be played below this value.
Impact Threshold	Float	Sound will forcefully play above this value.
Supress Warnings	Boolean	'true' if actor should not send warnings or errors to output log, 'false' if otherwise

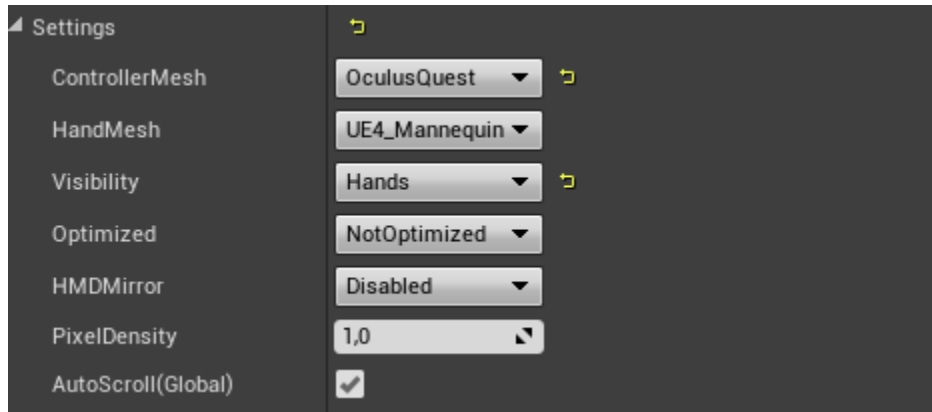
This is a very simple way of playing sound on collision. There are more advanced solutions on the Marketplace.



REQUIRED - Hit Events on the Mesh Component has to be enabled manually! If this is not set, the component will not work. This has to set up manually because this property can't be edited from BP.

OTHER FEATURES

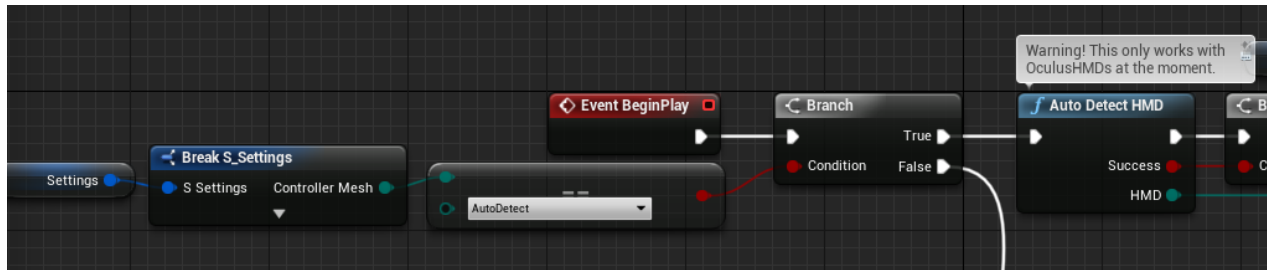
1. Settings



Settings		
Controller Mesh	E_HMD	Sets which mesh should be used for displaying controller
Hand Mesh	E_Hand	Sets which mesh should be used for displaying hand
Visibility	E_HACVisibility	Sets visibility of Hand and Controller
Optimized	E_Optimization	'Optimized' will load static mesh for controllers, 'NotOptimized' will load skeletal mesh for controllers
HMD Mirror	E_HMDMirrorMode	Sets which mirror mode should be used
Pixel Density	Float	A value to use with vr.PixelDensity
Auto Scroll	Boolean	'true' will allow Auto Scroll feature to work, 'false' will not
Start Transform	Transform	A transform to start the pawn at

The **Settings** variable contains the most major options in **BP_VRPawn** and **BP_VRPawn_GS**. These settings are passed down to **BPAC_HAC**.

Auto Detect



If **Controller Mesh** is set to AutoDetect, **CustomGM** will try to find which HMD is connected. If not, selected mesh will be used for display.

Warning: This only works with Oculus HMDs at the moment. Adding other HMDs are possible but we don't know how to identify them. **Get HMD Device Name** only returns SteamVR for other HMDs.

Please contact us if you know how to identify them.

Enable VR



This function makes sure VR is enabled and sets **Pixel Density**.

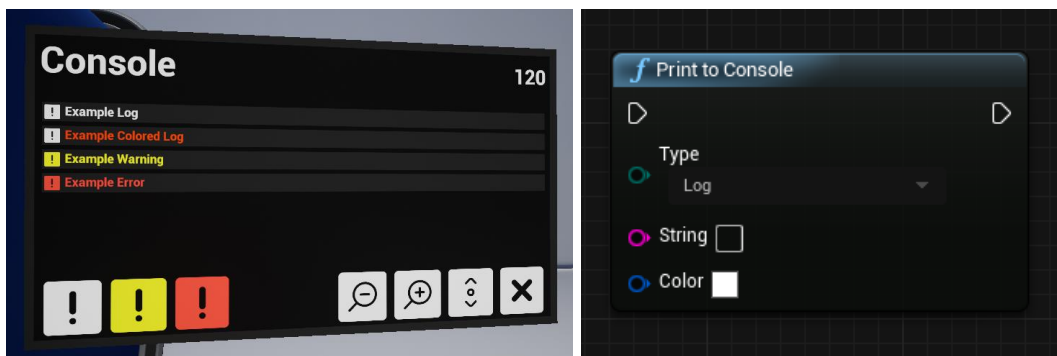
2. BP_VRPawn/BP_VRPawn_GS



Editable Variables		
Settings	S_Settings	Contains major settings
Hold Fade on Start	Boolean	'true' if camera should not fade in after SetupGraph is complete, 'false' otherwise
Fade Duration	Float	Teleport fade duration
Fade Color	Linear Color	A color to use when teleporting

These VR Pawns has been included to get things up and running out of the box. It can be used in most projects as is. In fact, we use a version of this pawn in our projects as well. This pawn shows how you can use the content with examples. **BP_VRPawn** only includes HAC logic, **BP_VRPawn_GS** includes Grab System logic as well.

3. VR_Console



Send text to console with specified color from any BP. Every print is sent to the **Output Log** as well.

Print to Console (Function) will print to **VR_Console** with the color specified but it will print with default color to **Output Log**.

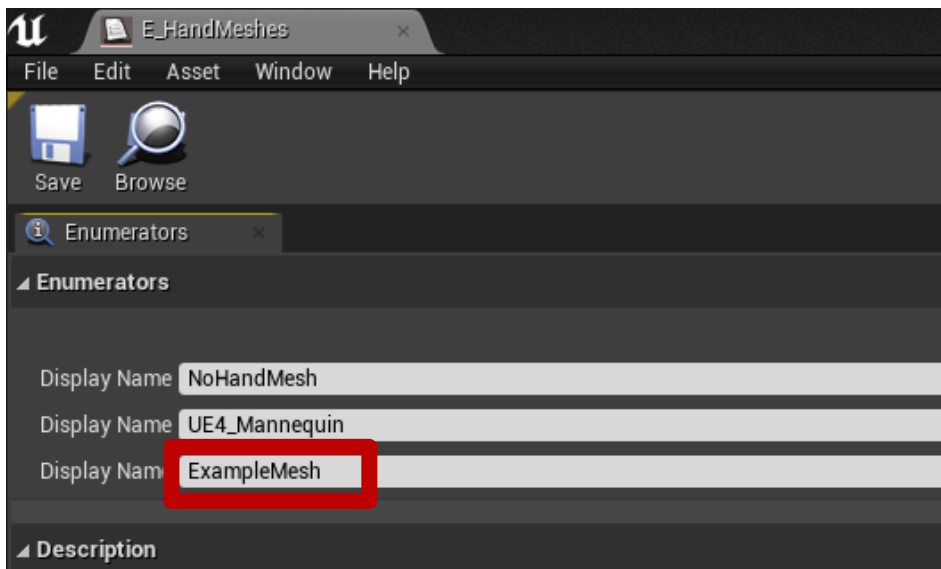
EXPANDING THE CONTENT

1. How to Add New Hand

Default UE4 Hand Skeletons

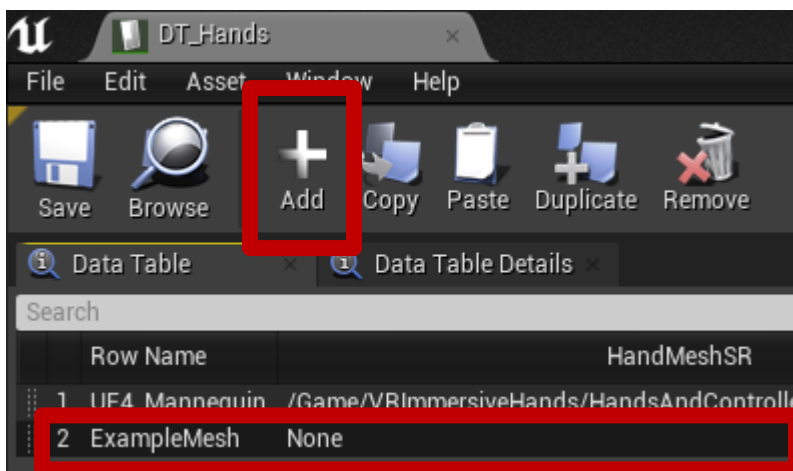
Any skeletal mesh that uses UE4's hand skeleton can be used in this content. To add a new hand;

- 1- Open **E_HandMeshes** and add a new enumerator.

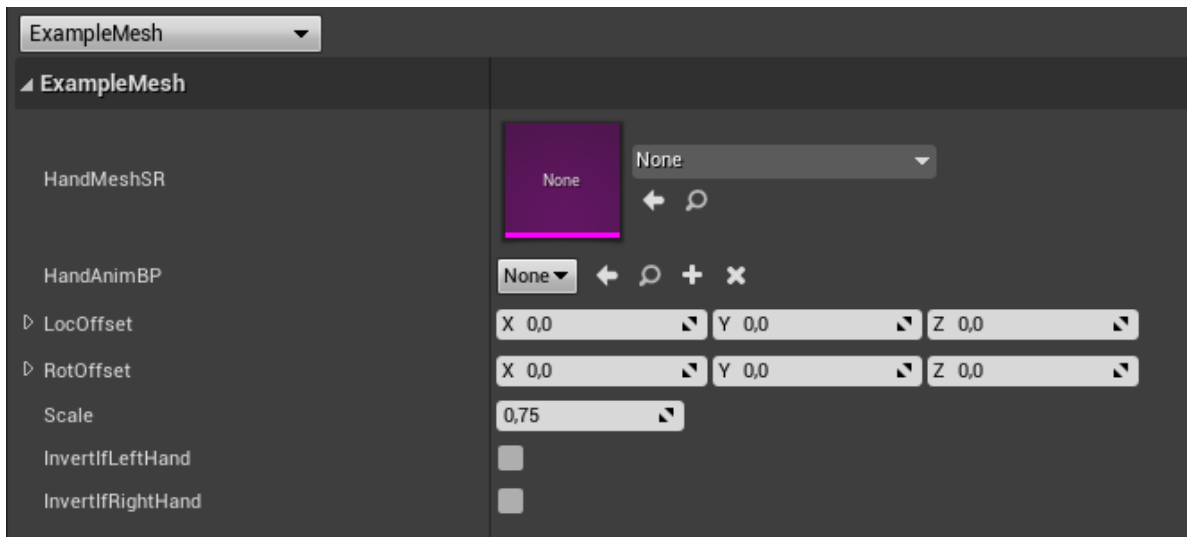


- 2- Open **DT_Hands** and add a new row.

Warning: This have to be same with the one entered in **E_HandMeshes**.



3- Fill out hand properties.



Editable Variables		
Hand Mesh SR	Skeletal Mesh SR	A skeletal mesh for displaying as hand
Hand Anim BP	Anim Instance SR	An anim instance for hand
Loc Offset	Vector	Offset to match UE4_Mannequin's location. Finding Hand Offset
Rot Offset	Rotator	Offset to match UE4_Mannequin's rotation. Finding Hand Offset
Scale	Float	A value to match UE4_Mannequin's 0.75 scale. Finding Hand Offset
Invert If Left Hand	Boolean	'true' if Y axis should be inverted on left hand, 'false' if not
Invert If Right Hand	Boolean	'true' if Y axis should be inverted on right hand, 'false' if not

Finding Hand Offset

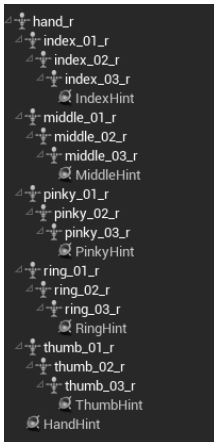
When using a different Skeletal Mesh for UE4's hand skeleton, it is not guaranteed that all of them will be exactly the same in size. Use **Loc Offset**, **Rot Offset** and **Scale** to match the new Skeletal Mesh to UE4_Mannequin. **Be aware**, if every animation looks good, great! You implemented a new hand without too much work. If they don't look so good, you will need to fine tune animations to look better.

Custom Hand Skeletons (Advanced)

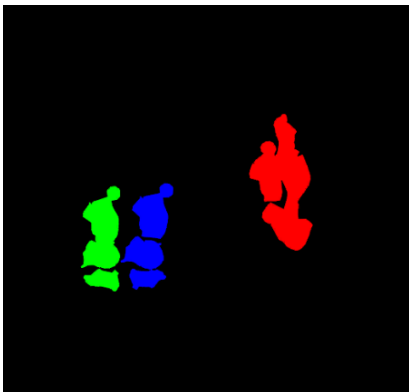
There is no easy way to implement a new skeleton to any system that relies on animations. We recommend you to take a look at Marketplace for default UE4 Hand Skeleton compatible meshes.

[Example](#). In any case, you can follow these steps to add custom hand skeleton.

1. Custom hand skeletons will need to retarget or recreate every animation in `"/VRImmersiveHands/HandsAndControllers/Hands/Mannequin/Anims"`. A fine-tune to every animation might be needed after retargeting. **Be aware**, this will take a long time. You can create a new Control Rig from the one we included for UE4 Hand Skeleton to speed up animation creation process.
2. Retarget **AnimBP_MannequinHand**.
3. Make sure to change the bone names in nodes to your hand.
4. Follow [Default UE4 Hand Skeletons](#).
5. Add hint sockets. Have a look at the provided hand.



6. Create a mask for every finger. Have a look at the provided hand.

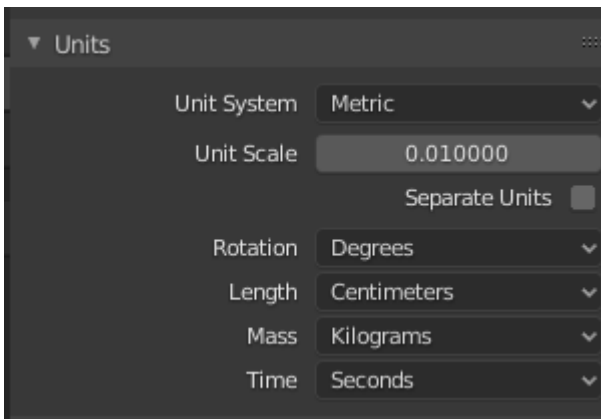


2. How to Add New Controller

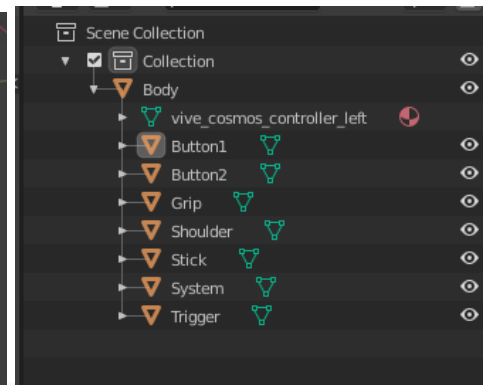
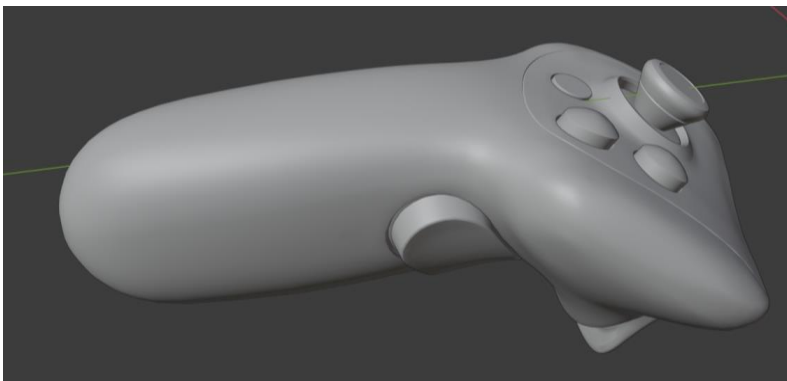
Importing the Mesh (Advanced)

Adding a new controller to the content is rather easy. To add a new controller mesh;

- 1- Find the 3D model for the controller. This usually can be found at manufacturer's website.
- 2- Import the 3D model to **Blender** and make sure scale is correct.
- 3- Set units as following.

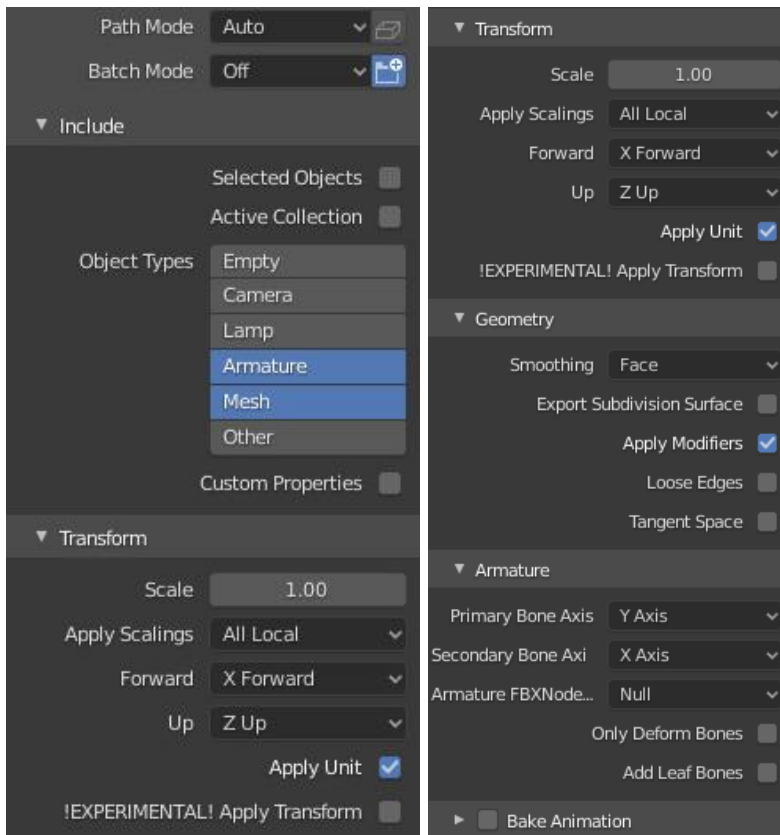


- 4- Separate, rename and parent buttons to Body. Rotate it so forward vector is X+. Name the face button closest to the hand as Button1 and further one as Button2.

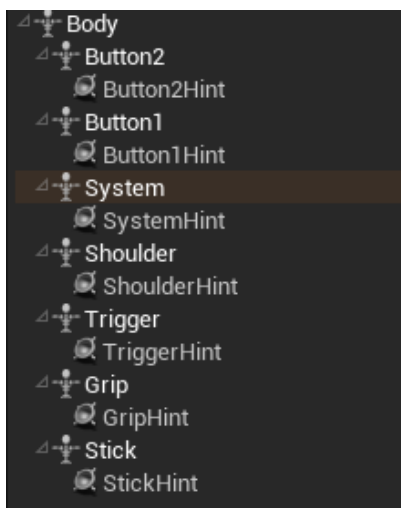


- 5- Set each button's pivot properly. Have a look at the Bones of the controller skeletal meshes included in the content for this. A good pivot placement will ease creating pressed button animation. There are tools to only change pivot in **Blender**. Exporting other skeletal meshes and having a look at them in **Blender** will help a lot. The 2 most important things are that you get the scale and pivot for the stick right.

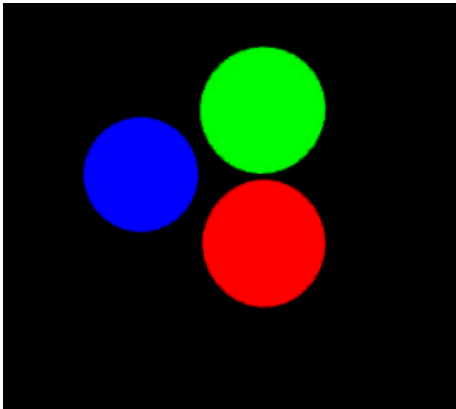
6- Export it.



- 7- Import it as Skeletal Mesh and leave Skeleton empty for the first one. If succesful, it will be imported as a Skeletal Mesh and a Skeleton will be created for it. You can use same Skeleton for Left and Right meshes.
- 8- Add hint sockets. Have a look at the provided controllers.

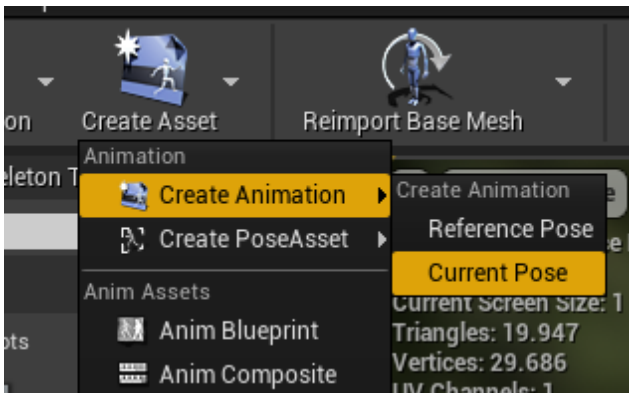


- 9- Create a mask for every button. Have a look at the provided controllers.



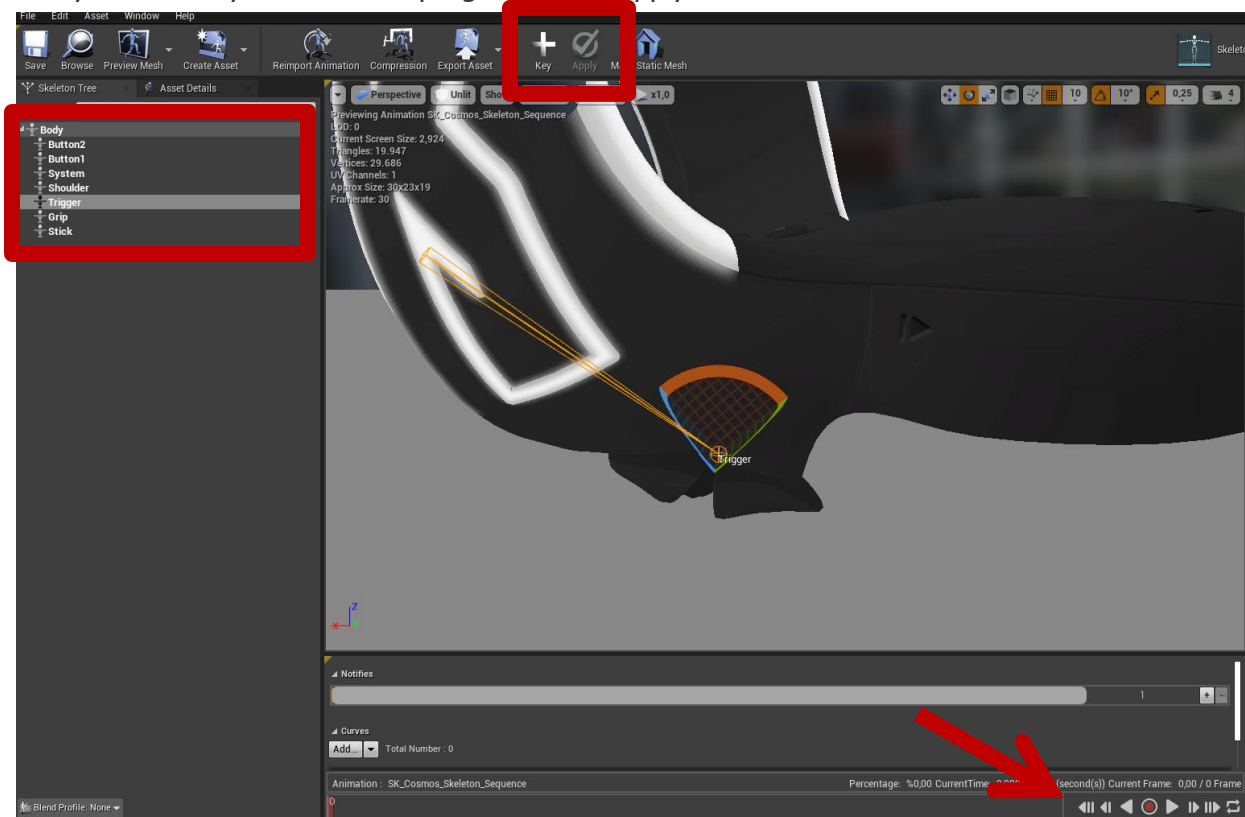
Creating Animation Assets (Advanced)

- 1- Open the Skeletal Mesh and Create Animation from Current Pose. Name it as Anim_XXX_Idle. Where XXX is the name of the controller.

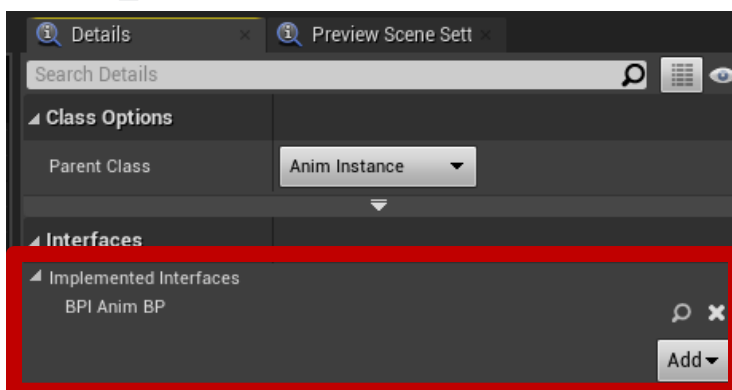


- 2- Duplicate Anim_XXX_Idle and name it as Anim_XXX_Pressed.

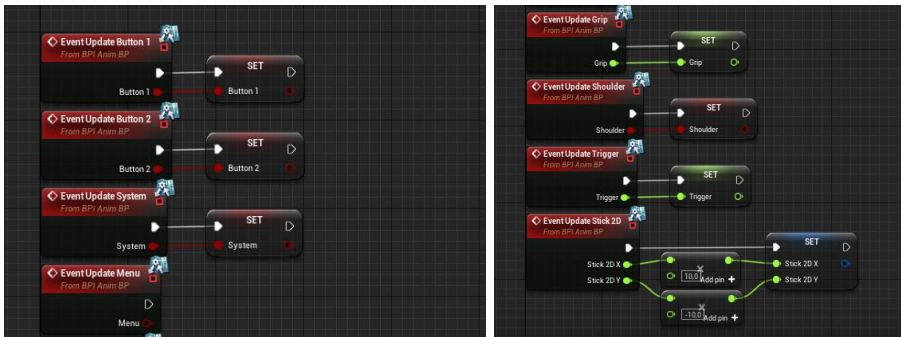
- 3- Open Anim_XXX_Pressed, click To Front on timeline and set each button's transform as though they are pressed. Have a look at other pressed anims for reference. Click Key and Apply after every button so you don't lose progress. Click Apply and Save when its done.



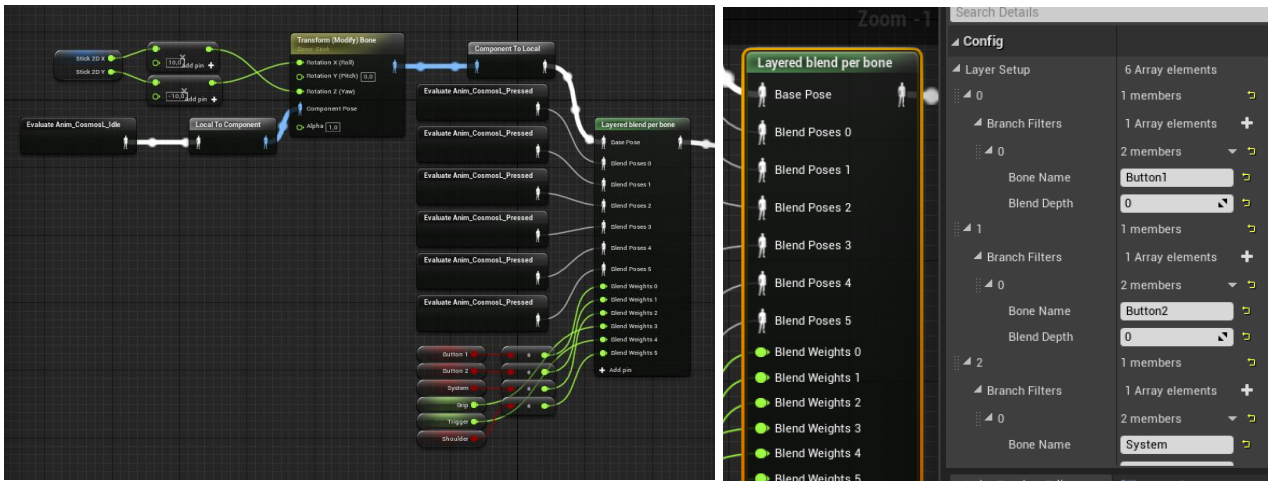
- 4- Create a Anim BP for each mesh.
 5- Add **BPI_AnimBP** to the Anim BP.



- 6- Copy the Interface Events from another AnimBP. Delete the one that your controller doesn't have.



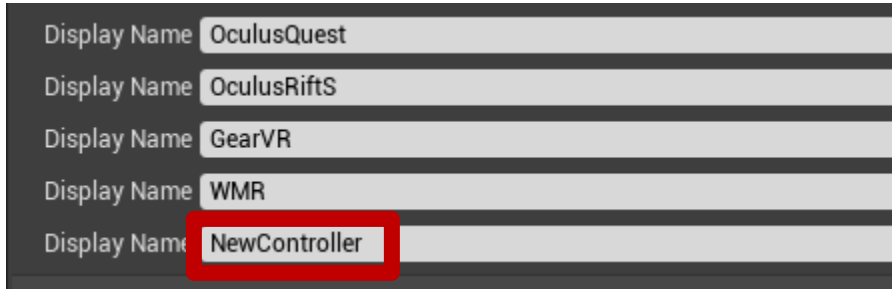
- 7- Copy the Anim Graph nodes from another AnimBP. Change Idle and Pressed anims with new ones. Delete the one that your controller doesn't have. Make sure bone names are correct in Layered Blend Per Bone.



- 8- Modify the Stick 2D part so that X=1, Y=0 tilts the stick on Y+ axis and X=0, Y=1 tilts the stick on X+ axis.

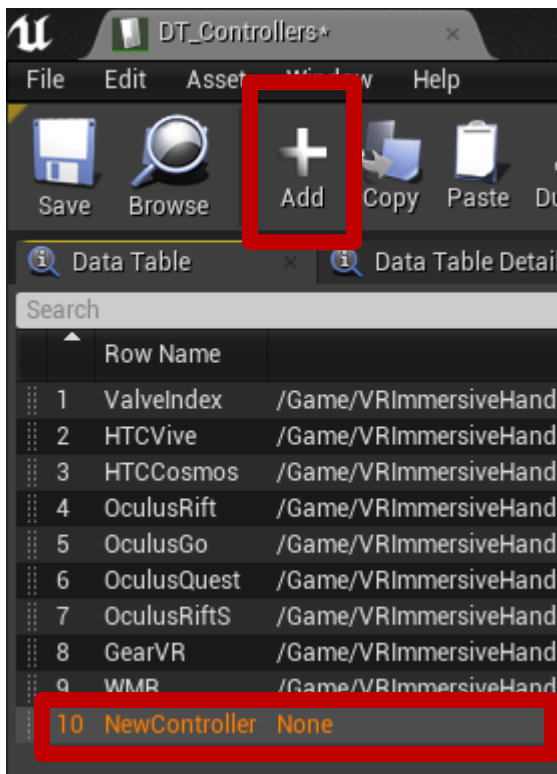
Adding it to the Data Table (Advanced)

- 1- Open **E_HMD** and add a new enumerator.

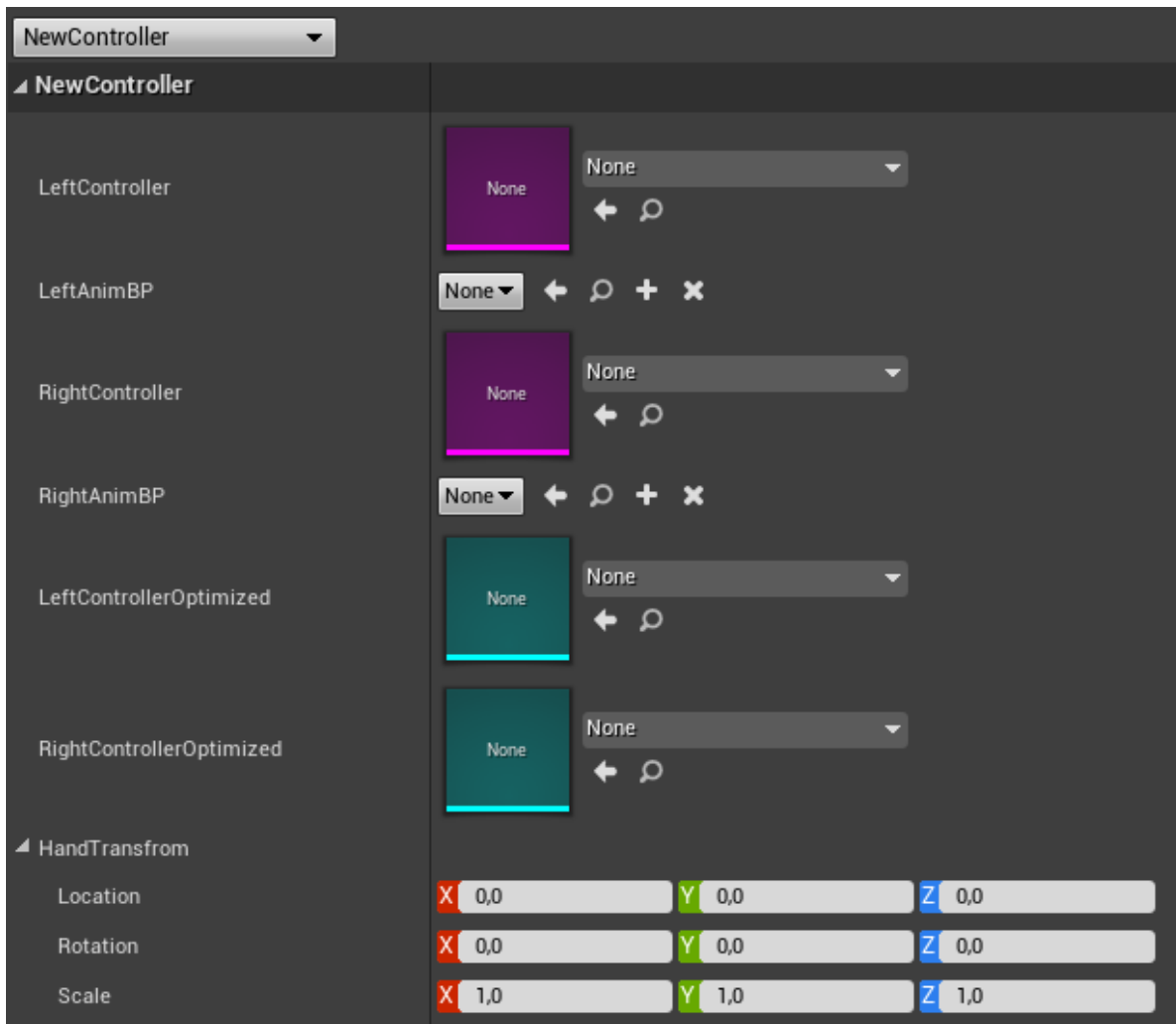


- 2- Open **DT_Controllers** and add a new row.

Warning: This have to be same with the one entered in **E_HMD**.



3- Fill out hand properties. We will come back to **Hand Transform**.

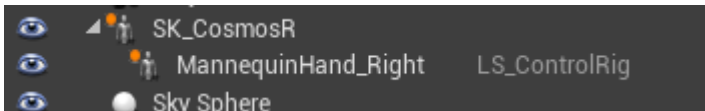


Editable Variables		
Left Controller	Skeletal Mesh SR	A skeletal mesh for displaying as left controller
Left Anim BP	Anim Instance SR	An anim instance for left controller
Right Controller	Skeletal Mesh SR	A skeletal mesh for displaying as right controller
Right Anim BP	Anim Instance SR	An anim instance for right controller
LC Optimized	Static Mesh SR	A static mesh for displaying as left controller
RC Optimized	Static Mesh SR	A static mesh for displaying as right controller
Hand Transform	Transform	Transform for UE4 hand to grab the controller Hand Animations

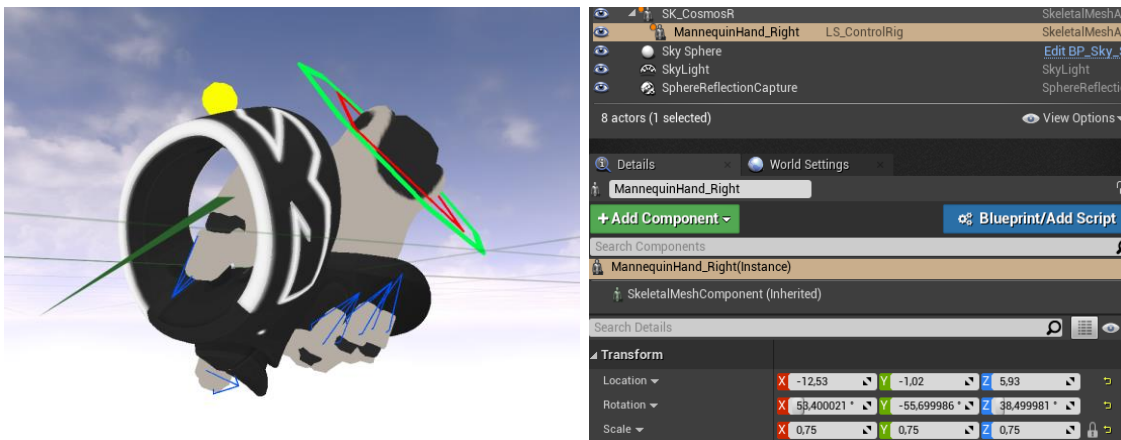
Hand Animations (Advanced)

Here comes the hard part. This part will take a long time and patience. If you want to use Control Rig, 4.25+ is required to export animations.

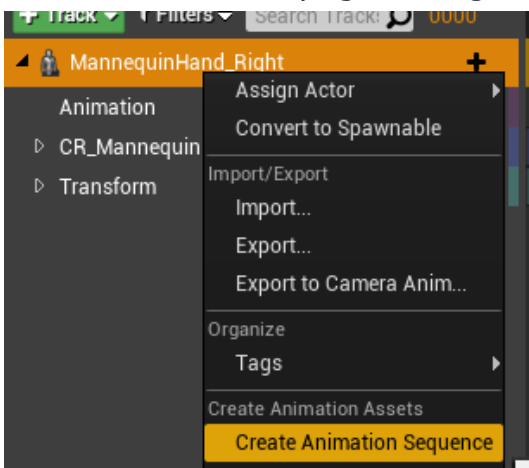
- 1- Open **Map_Setup** and put the new mesh in level.
- 2- Parent the hand to the controller and set hand's scale to 0,75.
- 3- Open **LS_ControlRig** and start creating animation. Animations should be 1 frame.
- 4- Parent the hand to the controller.



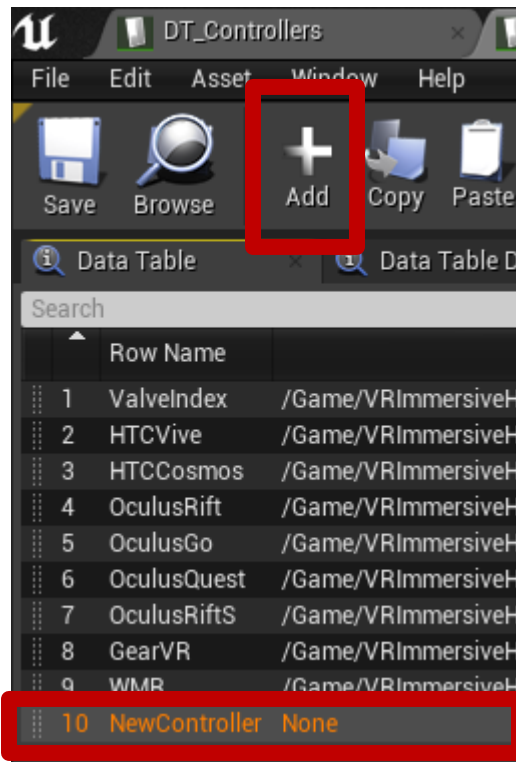
- 5- Find the most natural transform for hand to hold the controller and place the fingers. Key frequently in LS so you don't lose progress. Use this in [Adding it to the Data Table](#) Step 5.



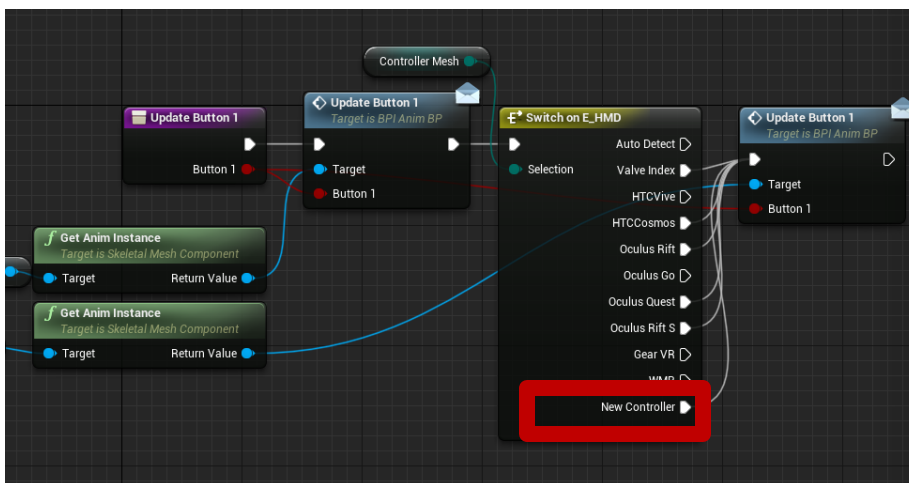
- 6- Place the fingers properly for Idle and each button. Have a look at other controllers for reference.
- 7- Save each animation by right clicking on MannequinHand_Right and clicking Create Anim Seq.



- 8- Open **DT_ControllerAnims**, add a new row and fill out animations. **Warning:** This have to be same with the one entered in **E_HMD**.



Update Button Functions (Advanced)



Go through each Update Button for each controller button.

Input Mappings

Check if your controller's inputs are mapped in "Project Settings>Input". If they are not, add them.

Note: Gear VR and Oculus Go inputs are deprecated. For HTC Cosmos, you will need Wave plugin from HTC for inputs.

MIGRATING



Click on the picture to go to Youtube video

1. Cleaning Up Before Migrating

DemoRoom folder is not needed for the content to work and we recommend you to delete it before migrating. Shut down the editor and delete **DemoRoom** folder manually. It should not give you any warning.

Warning: Editor sometimes thinks that it successfully deleted files if source control is enabled. But when the editor restarts, they would not be deleted. If this happens, make sure every file is not Read-Only and source control is disabled.

2. Migrating Input Mappings

We wanted this content to be an **Asset Pack** but it wasn't possible because we had to include input mappings as well.

- 1- Open **DefaultInput.ini** in **Config** folder.
- 2- Copy everything that starts with AxisConfig, ActionMappings and Axis Mappings. Then paste it in your project's **DefaultInput.ini** in **Config** folder.
- 3- You should now have every input mapping in your **Project Settings > Input**.

3. If You Only Want HAC (Optional)

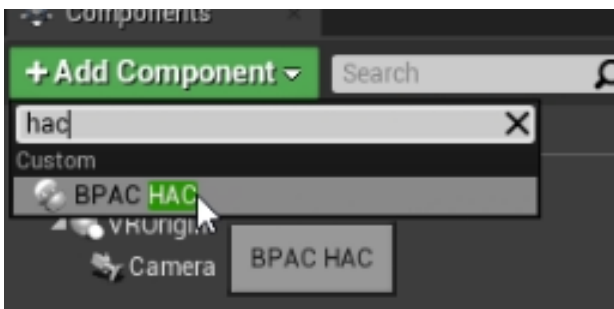
Follow these steps if you want to only migrate HAC. Otherwise skip this step.

- 1- Delete **Grab System** folder. It will warn you that some assets are using these files. Include them as well. It shouldn't ask you to **Force Delete**. If it does, shutdown the editor, delete Intermediate and Saved folders and try again.

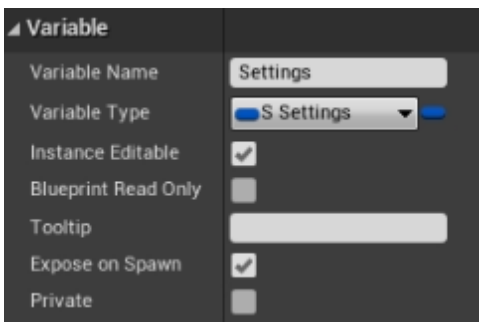
4. How to Use Your Own Pawn (Optional)

Follow these steps if you want to use your own pawn.

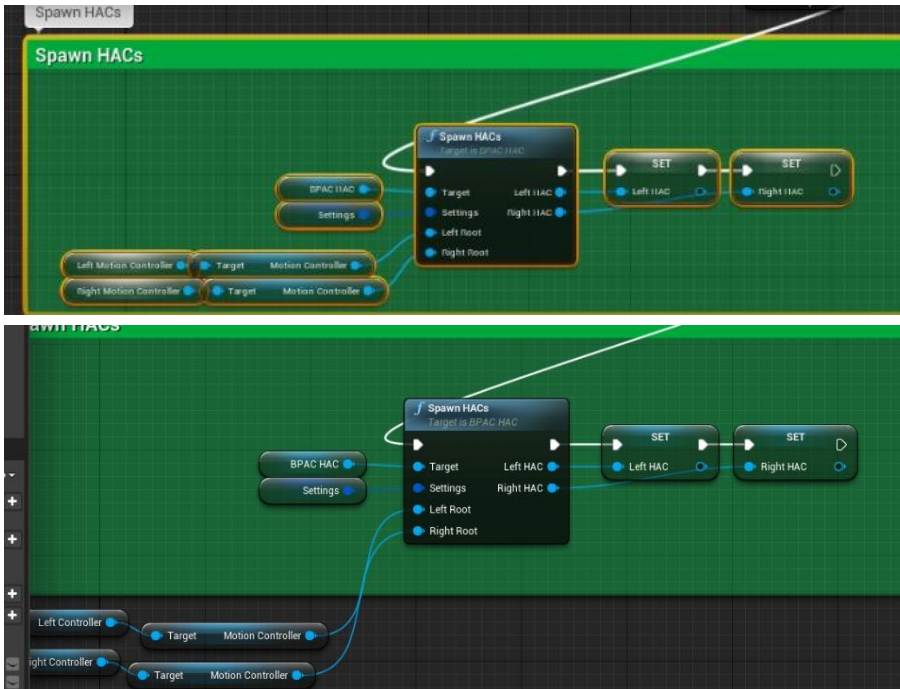
1. Add **BPAC_HAC** or **BPAC_HAC_GS** (if you migrated want to use GS) to your pawn. Don't use them together.



2. Add a **S_Settings** variable to your pawn, expose it on spawn and set its variables.



3. Copy **Spawn HAC** or **Spawn HAC_GS** logic from **BP_VRPawn** or **BP_VRPawn_GS** to your pawn and create missing variables.



4. Copy everything in **Input Graph** in **BP_VRPawn** or **BP_VRPawn_GS** to your BP. Only the teleport macro won't be copied.
5. **OPTIONAL** - Copy and paste AutoDetect logic in **BP_VRPawn** or **BP_VRPawn_GS** to your pawn if you want to use AutoDetect. Skip this step otherwise.

