htc

VIVE WAVE

Build Unreal Engine 5 OpenXR Applications on VIVE XR Devices

March. 2023

Revision History

Revision	Date	Description
1.0.0	May 2022	Formal release.
1.0.1	Jul 2022	- Supported UE5.0. - Added Hand Interaction and Wrist Tracker.
1.0.2	Aug 2022	- Supported UE OpenXREyeTracker plugin. - Added VIVE Focus3 controller model in sample zip. - Renamed plugin folder name to ViveOpenXRAndroid.
1.0.3	Sep 2022	- Supported Facial Tracking. - Renamed plugin name to VIVE OpenXR Plugin – Android.
1.0.5	Mar 2023	- Supported Passthrough Underlay. - Supported RenderDoc graphic debugger v1.25 for Vulkan. - Upgraded OpenXR Loader from 1.0.23 to 1.0.26. - Supported UE5.1 and do not support UE4.27 anymore.

1 Introduction

VIVE Wave runtime had already supported OpenXR standard on VIVE XR devices. Currently OpenXR supported devices are VIVE Focus3 and VIVE XR Elite. This guide will take VIVE Focus3 as target device to describe the statements.

In this guide, you will learn how to build an OpenXR app running on VIVE Focus3 through the following sections.

What you will learn in this guide:

- Download pre-built Unreal Engine 5.0 / 5.1 through Epic Games Launcher
- Package, install and run VIVE OpenXR application on VIVE Focus3
- Porting scene to use VIVE OpenXR - Android SDK

What Wave supports in this release version:

- HMD tracking pose and XR rendering
- Controller tracking pose, key input, and haptics
- Hand tracking pose and Hand Interaction
- Wrist Tracker
- Eye Tracking
- Facial Tracking
- Passthrough Underlay

Okay. Let’s get started to create your VIVE OpenXR content!

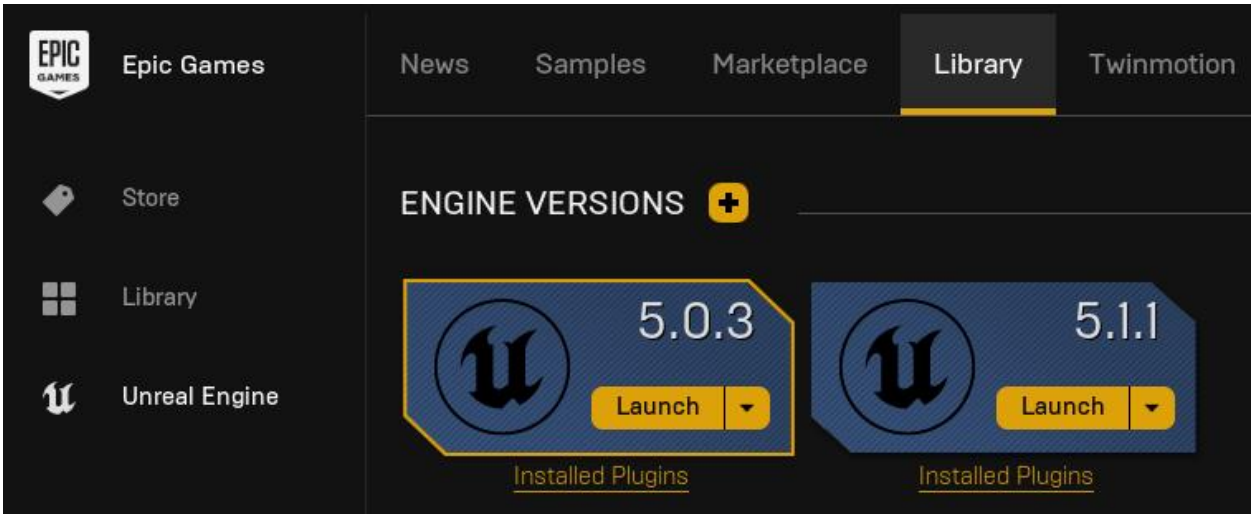
2 Download Pre-built Unreal Engine

2.1 Download Epic Games Launcher.

Download Epic Games Launcher here.

2.2 Download Unreal Engine by Epic Games Launcher

Navigate to ***Unreal Engine > Library*** and add ENGINE VERSIONS **5.0.3** or **5.1.1**.



3 VIVE OpenXR - Android SDK

3.1 VIVE OpenXR - Android Plug-In

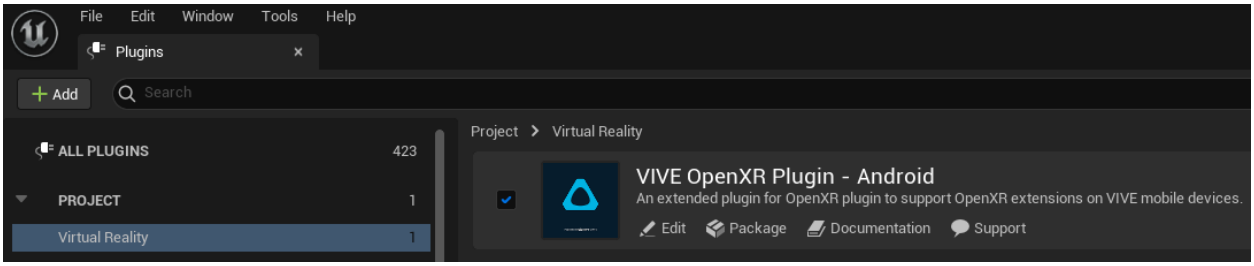
The **VIVE OpenXR - Android Plug-In** is at **<Where you unzip wave_openxr_plugin_<version>.zip>\ViveOpenXRAndroid**.

3.2 How to import VIVE OpenXR - Android Plug-In

Steps:

1. Copy and paste **ViveOpenXRAndroid** to *<Your Project Folder>\Plugins* such as *<Your Project Folder>\Plugins***ViveOpenXRAndroid** (Create a directory named "Plugins" if the project doesn't have.)
2. **[UE5.1 Only]** Must replace *<Your Prebuilt Engine Folder>\Epic Games\UE_5.1\Engine\Source\ThirdParty\Oculus\OculusOpenXRLoader\OculusOpenXRLoader\Lib\arm64-v8a**libopenxr_loader.so*** with *ViveOpenXRAndroid\Source\ViveOpenXRAndroidLoader\lib\android\arm64-v8a**libopenxr_loader.so*** because Oculus did not use the standard loader.
3. Double-click **<Your project>.uproject** to launch the project.

Then click *Unreal Engine Editor > Edit > Plugins*, the **VIVE OpenXR Plugin - Android** should be in the category *Project > VirtualReality*. Please relaunch the editor if the VIVE OpenXR Plugin - Android did not include in the Plugins list.



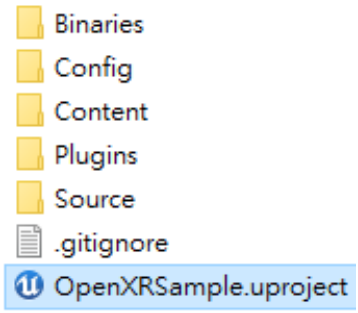
3.3 OpenXRSample

OpenXRSample is a sample project including scenes which let you experience our features such as input, hand tracking or other advanced features.

Unzip wave_openxr_sample_<version>.zip to get it. Please note that you should import VIVE OpenXR - Android Plug-In before packaging Android apps.

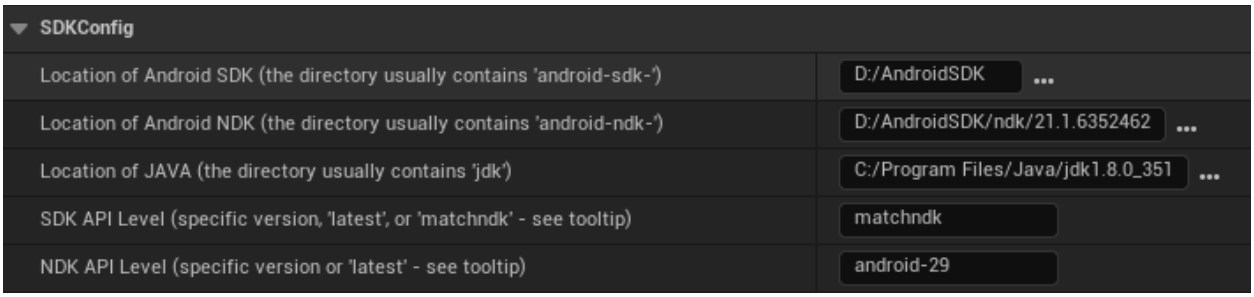
4 Package the Android Sample

Please make sure you already import the VIVE OpenXR - Android Plug-In. Then launch OpenXRSample**OpenXRSample.uproject**.



4.1 Project Settings of Android Environments

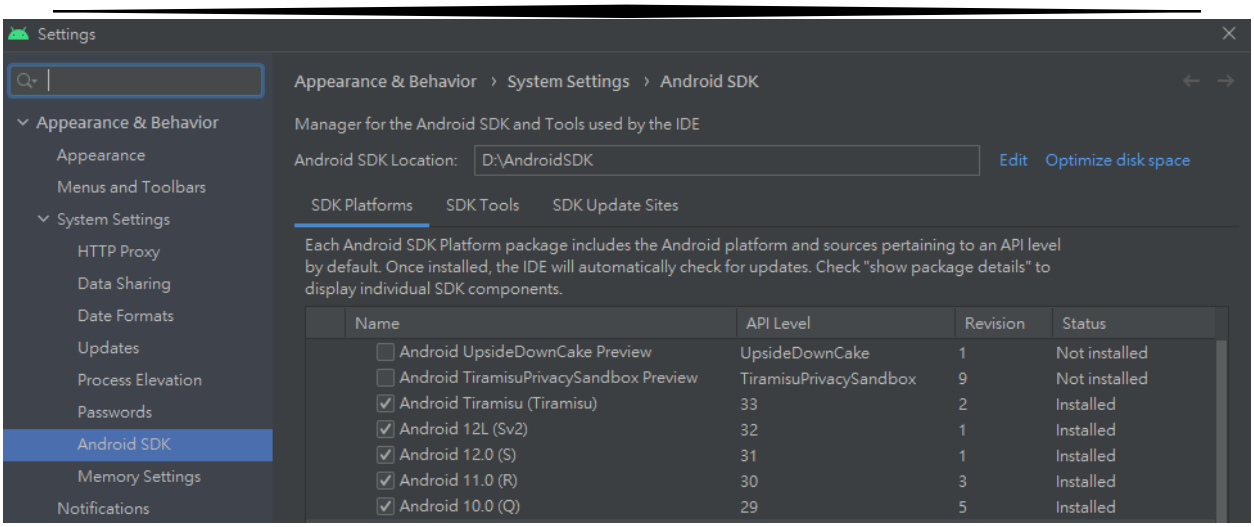
Click **Edit > Project Settings** to open the Project Settings window and navigate to **Platforms > Android SDK**.



You could refer [Setting up Android SDK and NDK for Unreal] to set up.

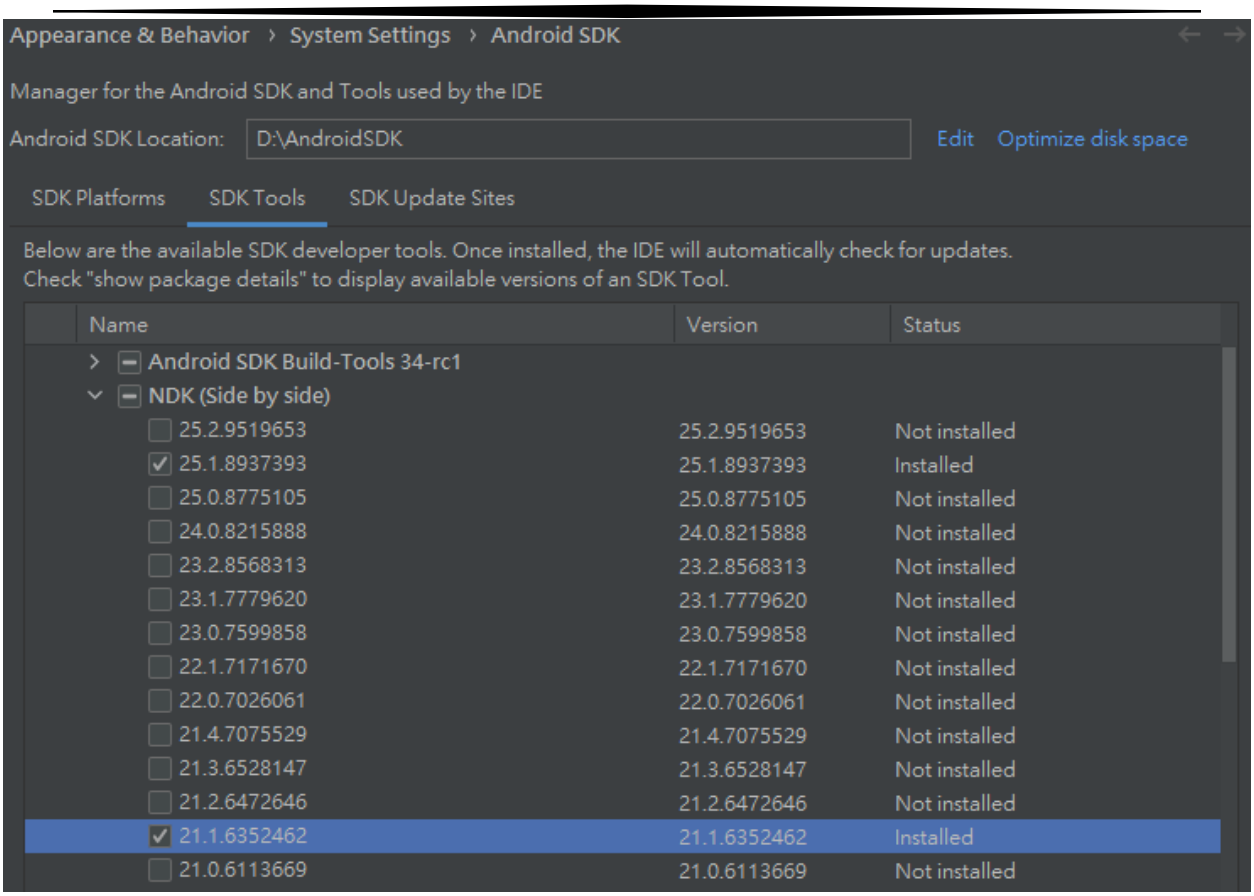
4.1.1 Location of Android SDK

Download Android Studio if you haven't yet. In Android Studio Editor, click *Tools > SDK Manager* and navigate to *Appearance & Behavior > System Settings > Android SDK*. The **Android SDK Location** is the path to be filled.



4.1.2 Location of Android NDK

Please fill your Android NDK(**21.1.6352462**) path to it for **UE5.0** and NDK(**25.1.8937393**) for **UE5.1**. The NDK could be downloaded in the Android SDK. Click the tab **SDK Tools** and select **Show Package Details** at bottom right of the page to see the options.

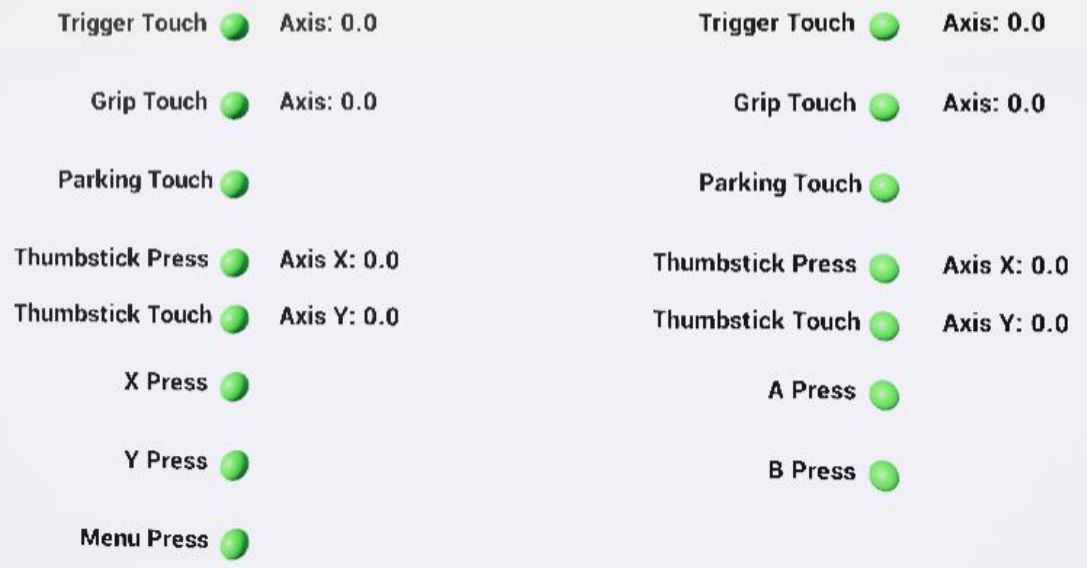


4.1.3 Location of JAVA

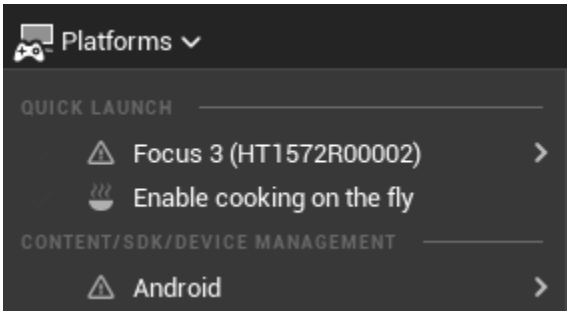
It is recommended to install jdk1.8.0.

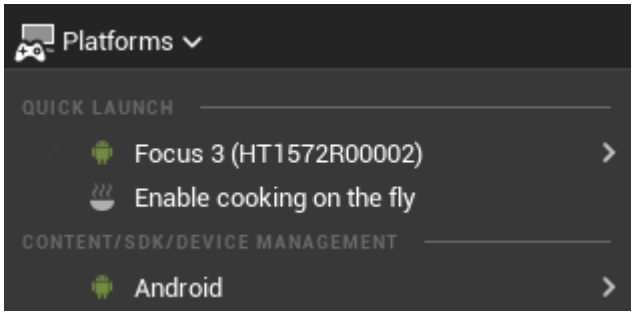
4.2 Package the ButtonTest to an Android APK

You will see the current level is **ButtonTest** which is located at *Content > ButtonTest > ButtonTest.umap*.



Click **Platforms > Android > Package Project** to package it. Click **Platforms > Refresh platform status** if there is no android robot icon show beside **Platforms > Android** but an exclamation mark.



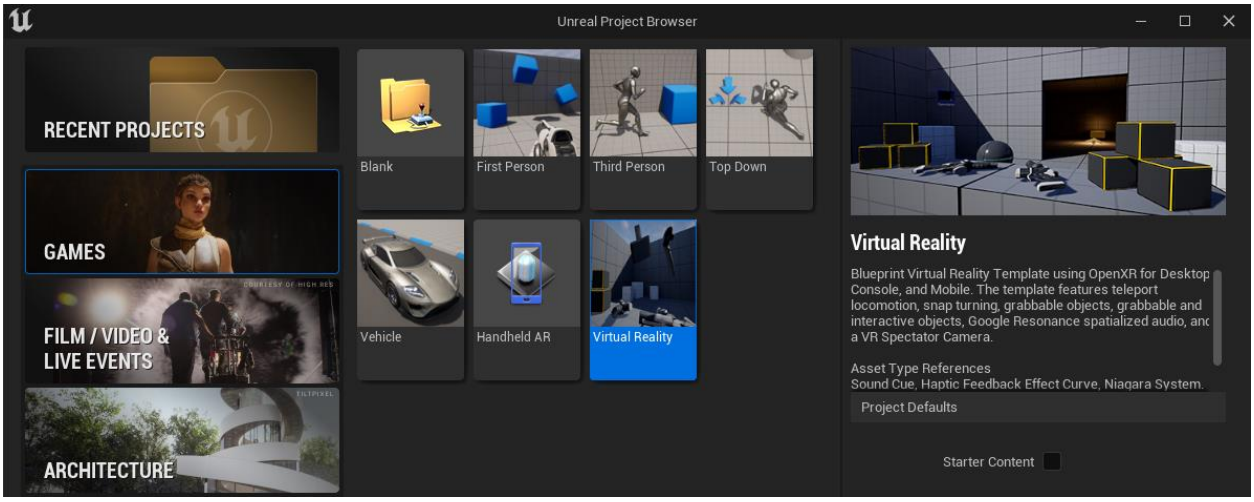


Then you will see the OpenXRSample-arm64.apk in the path *<path to OpenXRSample>/Android_ASTC/*. Execute **Install_OpenXRSample-arm64.bat** to install the android application. For more information, refer to Android Quick Start.

Make sure the controllers are paired well then launch the sample **OpenXRSample** in Focus3, you can see the pressing status and Axis value of controller buttons.

5 Porting scene to use VIVE OpenXR - Android

Unreal offered a Virtual Reality template project for starter (GAMES > Virtual Reality). You can migrate your scene to the project and refer the video below to get started.



[Creating AR and VR Projects with Unreal Engine | Webinar]

5.1 Import VIVE OpenXR Android Plug-In

Please refer chapter 3.2 above to see how to import the VIVE OpenXR - Android Plug-In.

5.2 Disable other Virtual Reality Plugin-Ins

Please keep **Oculus OpenXR**, **Oculus VR** and **SteamVR** disabled, and keep **OpenXR**, **OpenXREyeTracker**, **OpenXRHandTracking**, **VIVE OpenXR - Android** enabled.

5.3 Project Settings

5.3.1 Project Settings of Unreal Engine

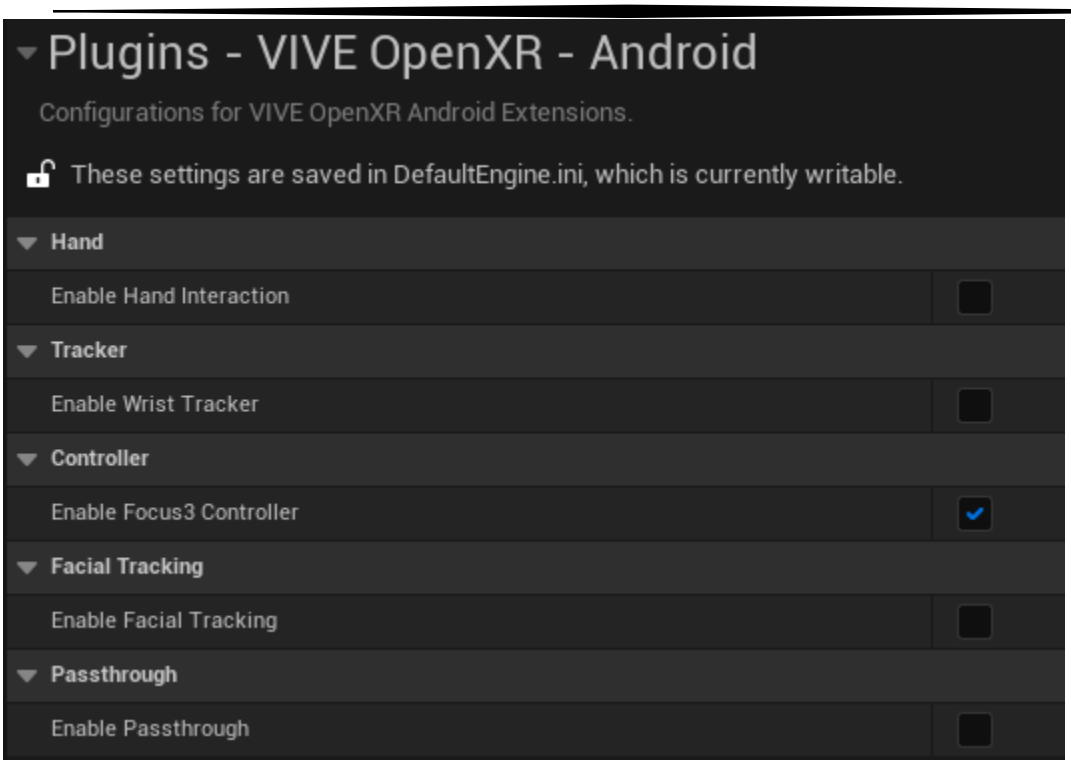
We assumed you migrate your scene to Virtual Reality project which Epic Games offered. You can refer other settings in Virtual Reality project that we didn't mention below.

Enable **Support Vulkan**. **(MUST)**

Disable **Show launch image** to get rid of the non-stereo splash while launching app.

5.3.2 Project Settings of VIVE OpenXR Android Plug-In

Click **Edit > Project Settings** to open the Project Settings window and navigate to **Plugins > VIVE OpenXR - Android**.



5.3.2.1 Enable Hand Interaction

Select this option to enable Hand Interaction feature. For more information, see chapter 6.2.

5.3.2.2 Enable Wrist Tracker

Select this option to enable Wrist Tracker feature. For more information, see chapter 6.3.

5.3.2.3 Enable Focus3 Controller

Select this option to support Focus3 controller. For more information, see chapter 5.4

5.3.2.4 Enable Facial Tracking

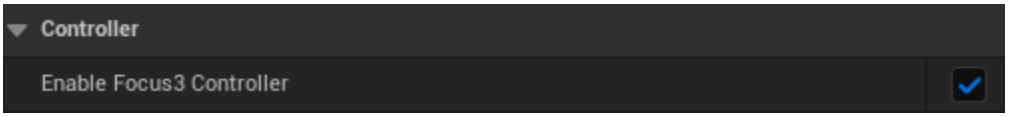
Select this option to support Facial Tracking feature. For more information, see chapter 6.5

5.3.2.5 Enable Passthrough

Select this option to support Passthrough feature. For more information, see chapter 6.6.

5.4 Input

To use the Focus3 controller inputs, please make sure **Project Settings > Plugins > VIVE OpenXR - Android > Enable Focus3 Controller** is enabled.

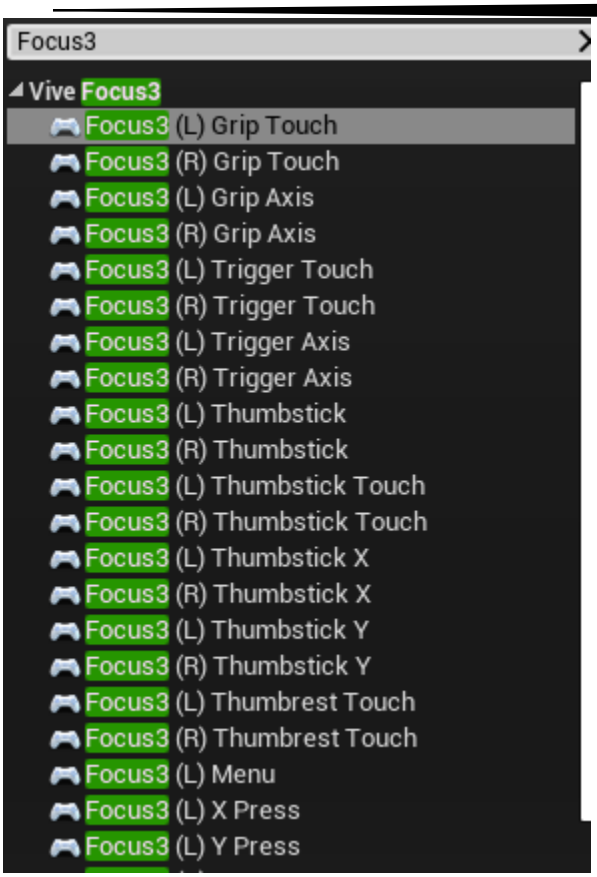


Then you can configure the **Action Mappings** and **Axis Mappings** in project settings for both UE5.0 and UE5.1. The alternative way for **UE5.1** is to use **Enhanced Input** because Action/Axis Mappings are deprecated in UE5.1.

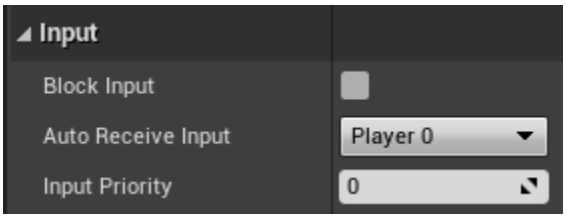
Refer to Input Overview for detailed information.

5.4.1 Setting Action and Axis Mappings in project settings

Go to **Project Settings > Engine > Input** to configure the **Action Mappings** and **Axis Mappings**. You can see the inputs of **VIVE Focus3** by typing “Focus3” in the action specifying window.



Note: you have to enable the **Auto Receive Input** option in your blueprints to retrieve inputs.



5.4.2 Enhanced Input (formal release in UE5.1)

Please refer Enhanced Input to use this feature. You can refer **All > Content > EnhancedInput > IMC_Keys** and **All > Content > ButtonTest > ButtonTest level blueprint** to see the input mapping sample in OpenXRSample project for **UE5.1**.

5.5 VIVE Focus 3 Controller Model

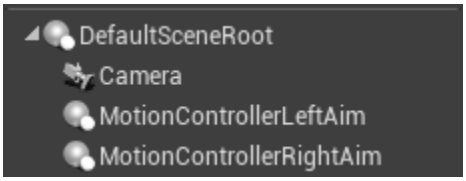
Because the OpenXR extension for controller model is still under discussion, we provide the VIVE Focus 3 controller asset for use now if the application needs.

5.5.1 Download the sample zip

Please download the sample zip and copy OpenXRSample/Content/Models/Controller to your project.

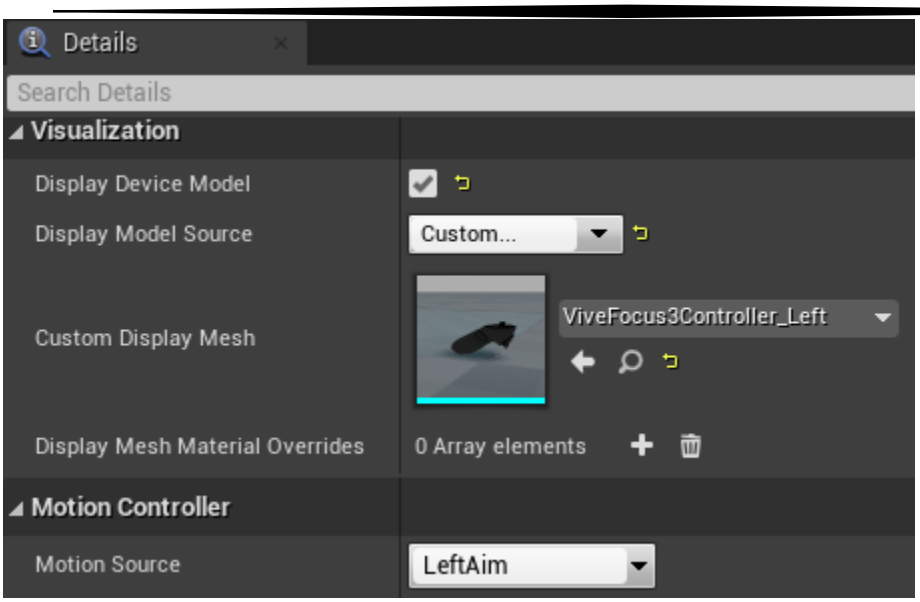
5.5.2 Setup the Controller Model in MotionController

Add MotionController components in your VRPawn if you didn't have. Such like:



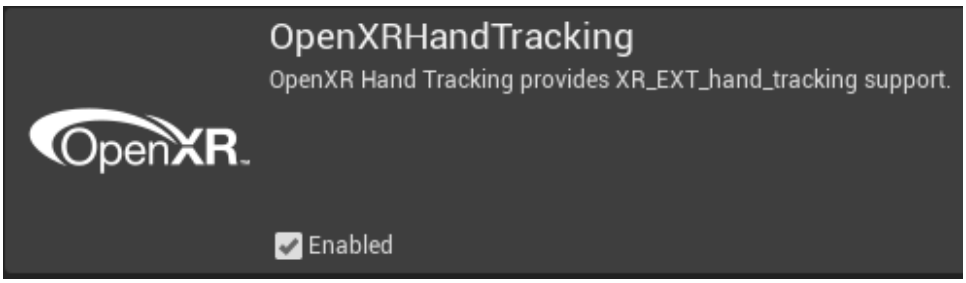
Click the MotionController component and check these settings:

1. Enable **Display Device Model.**
2. Set **Display Model Source** as Custom.
3. Select **Custom Display Mesh** as ViveFocus3Controller with the desired hand.
4. Select the **Motion Source** as LeftAim or RightAim.



6 Advanced Features

6.1 Hand Tracking



Please make sure the OpenXRHandTracking is enabled and refer to OpenXRSample/Content/NaturalHand/NaturalHand.umap

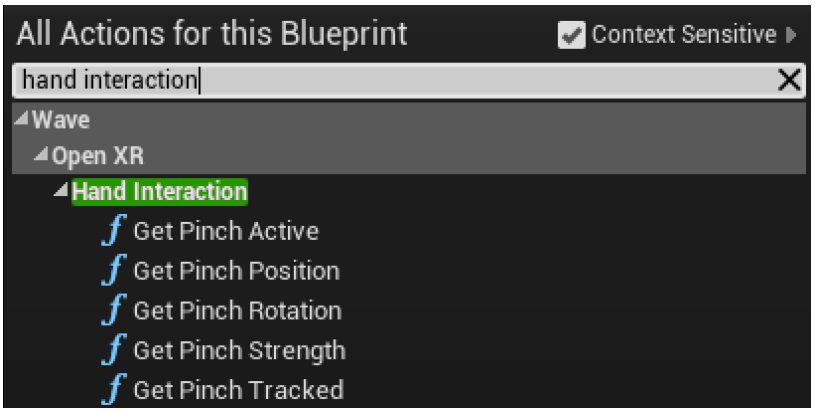
6.2 Hand Interaction

Use pinch motion to interact with Actors.

6.2.1 Project Setting

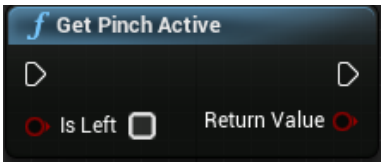
Click **Edit > Project Settings** to open the Project Settings window and navigate to **Plugins > VIVE OpenXR - Android**. Select **Enable Hand Interaction** to enable this feature.

6.2.2 Blueprint Function Libraries



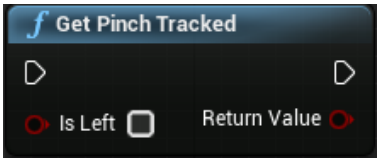
You can retrieve these pinch status of left hand if **Is Left** is true and the status of right hand if **Is Left** is false.

6.2.2.1 Get Pinch Active



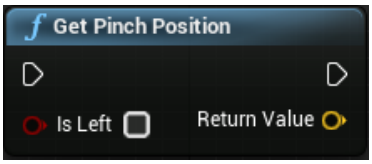
Checks if the pinch motion is active.

6.2.2.2 Get Pinch Tracked



Checks if the pinch motion is tracked.

6.2.2.3 Get Pinch Position



Retrieves the position of pinch origin of left or right hand.

6.2.2.4 Get Pinch Rotation

Retrieves the rotation of pinch origin of left or right hand.

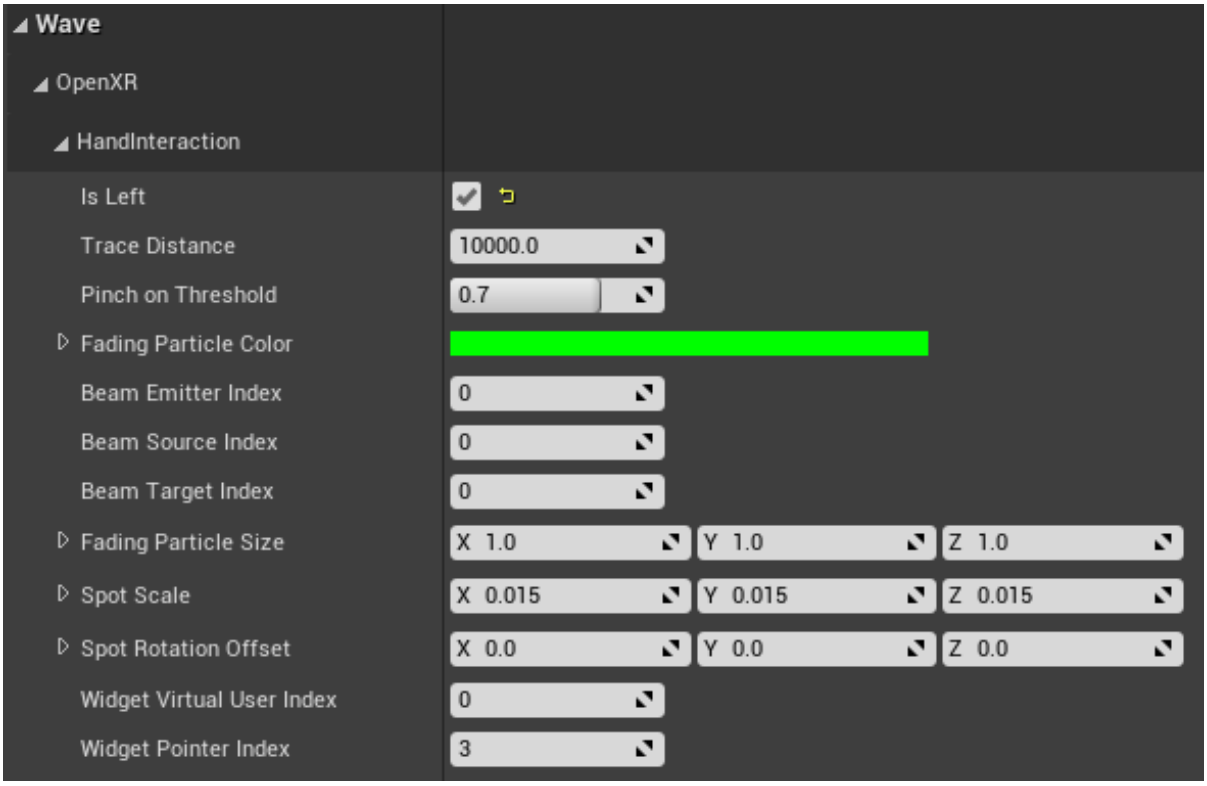
6.2.2.5 Get Pinch Strength

Retrieves the pinch strength of left or right hand. The return value should be [0, 1]. 1 means the index finger is closed to thumb very much.

6.2.3 Wave Hand Pointer

Wave Hand Pointer is a SceneComponent which use the Hand Interaction feature. Wave Hand Pointer makes you easy to interact with Actors by hand.

The detail of Wave Hand Pointer:



Please refer OpenXRSample/Content/NaturalHand/NaturalHand.umap to see the sample.

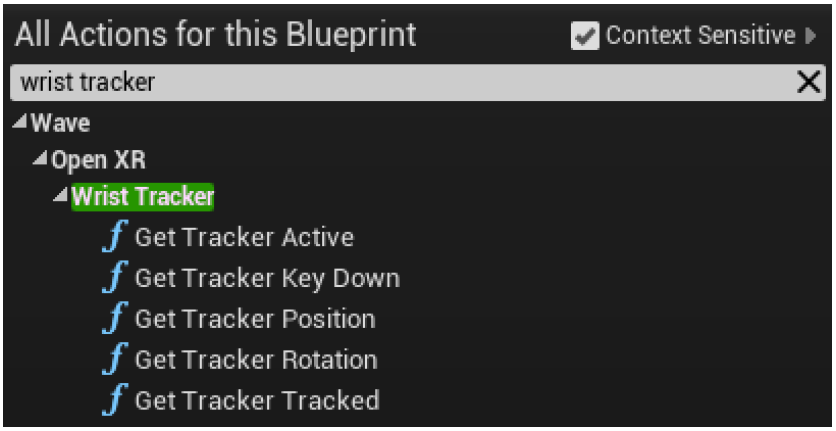
6.3 Wrist Tracker

Retrieve the status of Wrist Tracker.

6.3.1 Project Settings

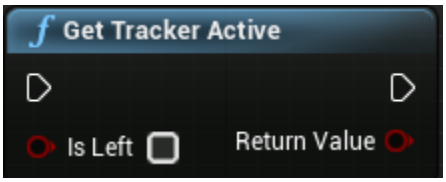
Click **Edit > Project Settings** to open the Project Settings window and navigate to **Plugins > VIVE OpenXR - Android**. Select **Enable Wrist Tracker** to enable this feature.

6.3.2 Blueprint Function Libraries



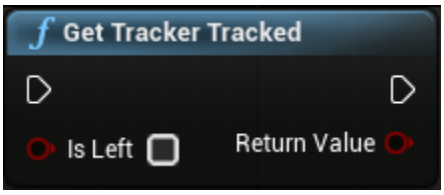
You can retrieve these tracker status of left or right device by setting **Is Left**. The **L / R** light mark the device is left or right. The L light will be on if the tracker is the left device.

6.3.2.1 Get Tracker Active



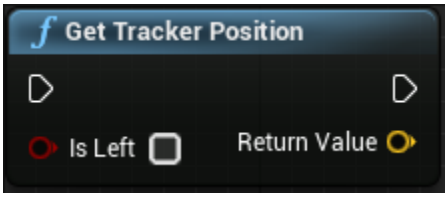
Checks if the tracker is active. (The device is connected or not.)

6.3.2.2 Get Tracker Tracked



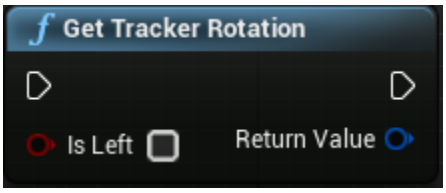
Checks if the tracker is tracked.

6.3.2.3 Get Tracker Position



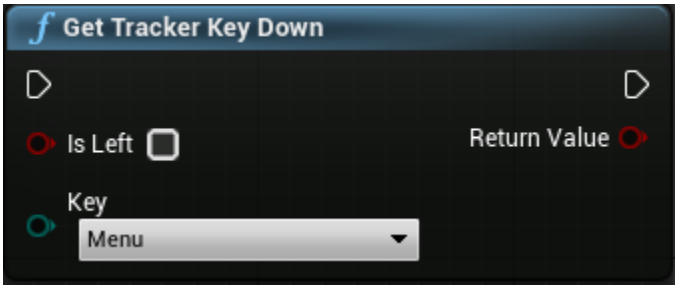
Retrieves the position of the left or right tracker.

6.3.2.4 Get Tracker Rotation



Retrieves the rotation of the left or right tracker.

6.3.2.5 Get Tracker Key Down



There are two buttons for each tracker. They are **Menu** and **Primary**.

Checks if these buttons pressing or not (releasing).

Note: You cannot retrieve the key status of Menu on the right device since it's a reserved key.

6.3.3 Key event of Tracker input

The Get Tracker Key Down Blueprint Function is a polling way to get key status. If you want to listen one key event when the button had been pressed or released. You can register your input action event and add Tracker inputs in it:

Click **Edit > Project Settings** to open the Project Settings window and navigate to **Engine > Input**.

There are four inputs belongs to tracker, two for each tracker:

Tracker (L) Menu, Tracker (L) X Press, Tracker (R) System and Tracker (R) A Press.

Note: You cannot retrieve the Tracker (R) System key status since it's a reserved key.

For example, I register the Input Action TriggerRight and add the **Tracker (R) A Press** in it, then I can listen the key event by InputAction TriggerRight.

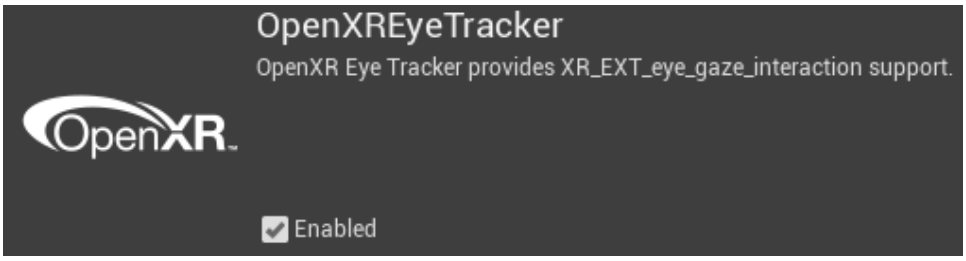
The alternative way for UE5.1, you can use Enhanced Input to listen these keys as well. For more information, see chapter 5.4.2

For more information, refer to InteractiveExperiences-Input.

6.4 Eye Tracking

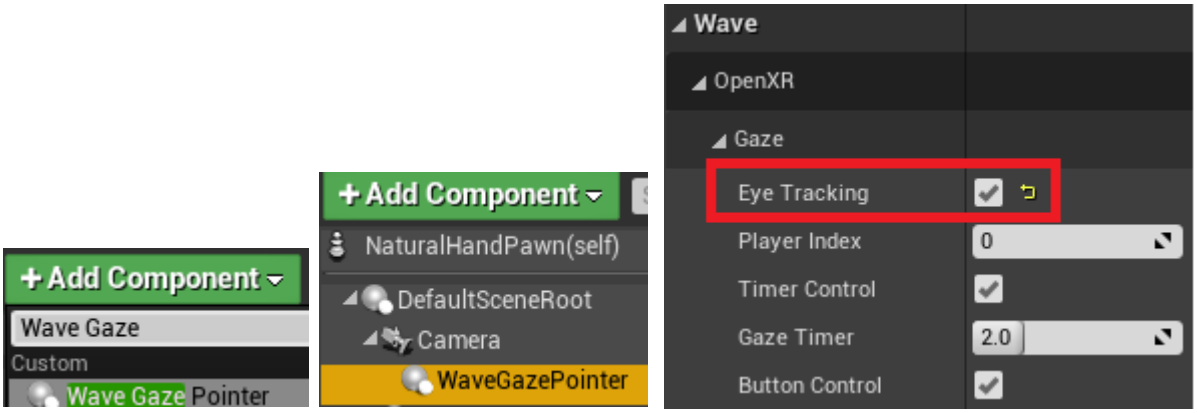
Unreal engine provides the **Eye Tracking** feature by enabling the **OpenXREyeTracker** plugin from *Plugin > Built-In > Virtual Reality > OpenXREyeTracker*.

After enabled the **Eye Tracking** feature, you can access the **Eye Tracking** data from UEyeTrackerFunctionLibrary interface.



6.4.1 Eye Gaze

VIVE OpenXR Android plugin provides a SceneComponent named **WaveGazePointer** which can use the **Eye Tracking** data as gaze direction. By using the **WaveGazePointer** with **Eye Tracking** enabled, you can see a moving gaze pointer accompanying with your sight. We demonstrate the **Eye Gaze** which uses **WaveGazePointer** in the sample *OpenXRSample > Content > NaturalHand*.



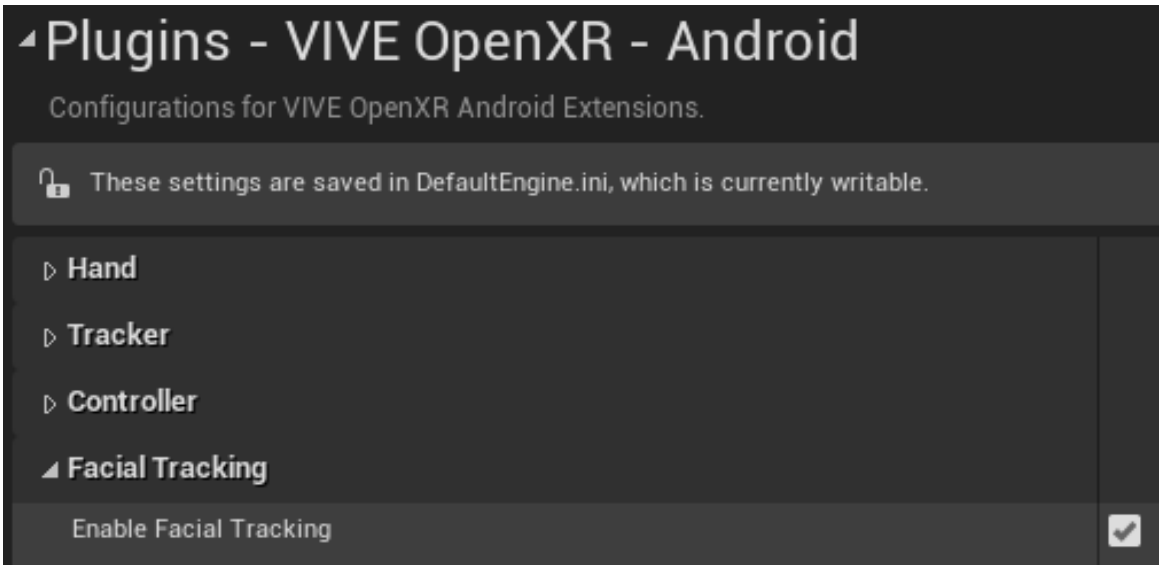
If you don’t enable the **Eye Tracking** option, the gaze pointer will be fixed at the center of screen.

6.5 Facial Tracking

VIVE OpenXR Android plugin supports the OpenXR 12.69. XR_HTC_facial_tracking specification.

6.5.1 Project Settings

Before using the **Facial Tracking** you have to enable the feature in *Project Settings > Plugins – VIVE OpenXR – Android > Facial Tracking*.



6.5.2 Blueprint Function Library

VIVE OpenXR Android plugin provides the Blueprint Function Library of **Facial Tracking** in ViveOpenXRAndroid > Source > ViveOpenXRAndroidFacialTracking > Public ViveFacialExpressionEnums.h and ViveOpenXRAndroidFacialTrackingFunctionLibrary.h.

```
UENUM(BlueprintType, Category = "ViveOpenXRAndroid|OpenXR|FacialTracking")
enum class EXrFacialTrackingType : uint8 {
	None = 0  UMETA(Hidden),
	Eye = 1,
	Lip = 2
};
```

```
/** The avatar's eye relative blend shape.*/
UENUM(BlueprintType, Category = "ViveOpenXRAndroid|OpenXR|FacialTracking")
enum class EEyeShape :uint8 {
      Eye_Left_Blink = 0   UMETA(DisplayName = "Left_Blink"),
      Eye_Left_Wide = 1    UMETA(DisplayName = "Left_Wide"),
      Eye_Left_Right = 2   UMETA(DisplayName = "Left_In"),
      Eye_Left_Left = 3    UMETA(DisplayName = "Left_Out"),
      Eye_Left_Up = 4      UMETA(DisplayName = "Left_Up"),
      Eye_Left_Down = 5    UMETA(DisplayName = "Left_Down"),
      Eye_Right_Blink = 6  UMETA(DisplayName = "Right_Blink"),
      Eye_Right_Wide = 7   UMETA(DisplayName = "Right_Wide"),
      Eye_Right_Right = 8  UMETA(DisplayName = "Right_Out"),
      Eye_Right_Left = 9   UMETA(DisplayName = "Right_In"),
      Eye_Right_Up = 10    UMETA(DisplayName = "Right_Up"),
      Eye_Right_Down = 11  UMETA(DisplayName = "Right_Down"),
      Eye_Frown = 12       UMETA(DisplayName = "Reserved"),
      Eye_Left_Squeeze = 13  UMETA(DisplayName = "Left_Squeeze"),
      Eye_Right_Squeeze = 14 UMETA(DisplayName = "Right_Squeeze"),
      Max = 15               UMETA(Hidden),
      None = 63              UMETA(DisplayName = "None"),
};

void GetIsFacialTrackingEnabled(bool& result)
```

- Checks if **Facial Tracking** is enabled in Project Settings.

```
bool CreateFacialTracker(EXrFacialTrackingType trackingType)
```

- Creates a facial tracker before retrieving the **Facial Tracking** data.

```
bool bool DestroyFacialTracker(EXrFacialTrackingType trackingType)
```

- Destroys a facial tracker when not using **Facial Tracking** anymore.

```
bool GetEyeFacialExpressions(bool& isActive, TMap<EEyeShape, float>& blendshapes)
```

- Retrieves the weightings of eye's blend shape.

```
bool GetLipFacialExpressions(bool& isActive, TMap<ELipShape, float>& blendshapes)
```

- Retrieves the weightings of lip blend shape.

6.5.3 Sample code

You can find the sample code ViveOpenXRAndroid > Source > ViveOpenXRAndroidFacialTracking > Private - FT_AvatarSample.cpp demonstrates the usage of **Facial Tracking** blueprint function library.

```
/** The prediction result relative blend shape.*/
UENUM(BlueprintType, Category = "ViveOpenXRAndroid|OpenXR|FacialTracking")
enum class ELipShape :uint8 {
  Jaw_Right = 0                UMETA(DisplayName = "Jaw_Right"),
  Jaw_Left = 1                 UMETA(DisplayName = "Jaw_Left"),
  Jaw_Forward = 2              UMETA(DisplayName = "Jaw_Forward"),
  Jaw_Open = 3                 UMETA(DisplayName = "Jaw_Open"),
  Mouth_Ape_Shape = 4          UMETA(DisplayName = "Mouth_Ape_Shape"),
  Mouth_Upper_Right = 5        UMETA(DisplayName = "Mouth_Upper_Right"),
  Mouth_Upper_Left = 6         UMETA(DisplayName = "Mouth_Upper_Left"),
  Mouth_Lower_Right = 7        UMETA(DisplayName = "Mouth_Lower_Right"),
  Mouth_Lower_Left = 8         UMETA(DisplayName = "Mouth_Lower_Left"),
  Mouth_Upper_Overturn = 9     UMETA(DisplayName = "Mouth_Upper_Overturn"),
  Mouth_Lower_Overturn = 10    UMETA(DisplayName = "Mouth_Lower_Overturn"),
  Mouth_Pout = 11              UMETA(DisplayName = "Mouth_Pout"),
  Mouth_Smile_Right = 12       UMETA(DisplayName = "Mouth_Smile_Right"),
  Mouth_Smile_Left = 13        UMETA(DisplayName = "Mouth_Smile_Left"),
  Mouth_Sad_Right = 14         UMETA(DisplayName = "Mouth_Sad_Right"),
  Mouth_Sad_Left = 15          UMETA(DisplayName = "Mouth_Sad_Left"),
  Cheek_Puff_Right = 16        UMETA(DisplayName = "Cheek_Puff_Right"),
  Cheek_Puff_Left = 17         UMETA(DisplayName = "Cheek_Puff_Left"),
  Cheek_Suck = 18              UMETA(DisplayName = "Cheek_Suck"),
  Mouth_Upper_UpRight = 19     UMETA(DisplayName = "Mouth_Upper_UpRight"),
  Mouth_Upper_UpLeft = 20      UMETA(DisplayName = "Mouth_Upper_UpLeft"),
  Mouth_Lower_DownRight = 21   UMETA(DisplayName = "Mouth_Lower_DownRight"),
  Mouth_Lower_DownLeft = 22    UMETA(DisplayName = "Mouth_Lower_DownLeft"),
  Mouth_Upper_Inside = 23      UMETA(DisplayName = "Mouth_Upper_Inside"),
  Mouth_Lower_Inside = 24      UMETA(DisplayName = "Mouth_Lower_Inside"),
  Mouth_Lower_Overlay = 25     UMETA(DisplayName = "Mouth_Lower_Overlay"),
  Tongue_LongStep1 = 26        UMETA(DisplayName = "Tongue_LongStep1"),
  Tongue_Left = 27             UMETA(DisplayName = "Tongue_Left"),
  Tongue_Right = 28            UMETA(DisplayName = "Tongue_Right"),
  Tongue_Up = 29               UMETA(DisplayName = "Tongue_Up"),
  Tongue_Down = 30             UMETA(DisplayName = "Tongue_Down"),
  Tongue_Roll = 31             UMETA(DisplayName = "Tongue_Roll"),
  Tongue_LongStep2 = 32        UMETA(DisplayName = "Tongue_LongStep2"),
  Tongue_UpRight_Morph = 33    UMETA(DisplayName = "Tongue_UpRight_Morph"),
  Tongue_UpLeft_Morph = 34     UMETA(DisplayName = "Tongue_UpLeft_Morph"),
  Tongue_DownRight_Morph = 35  UMETA(DisplayName = "Tongue_DownRight_Morph"),
  Tongue_DownLeft_Morph = 36   UMETA(DisplayName = "Tongue_DownLeft_Morph"),
  Max = 37                     UMETA(Hidden),
  None = 63                    UMETA(DisplayName = "None"),
};
```

6.6 Passthrough Underlay

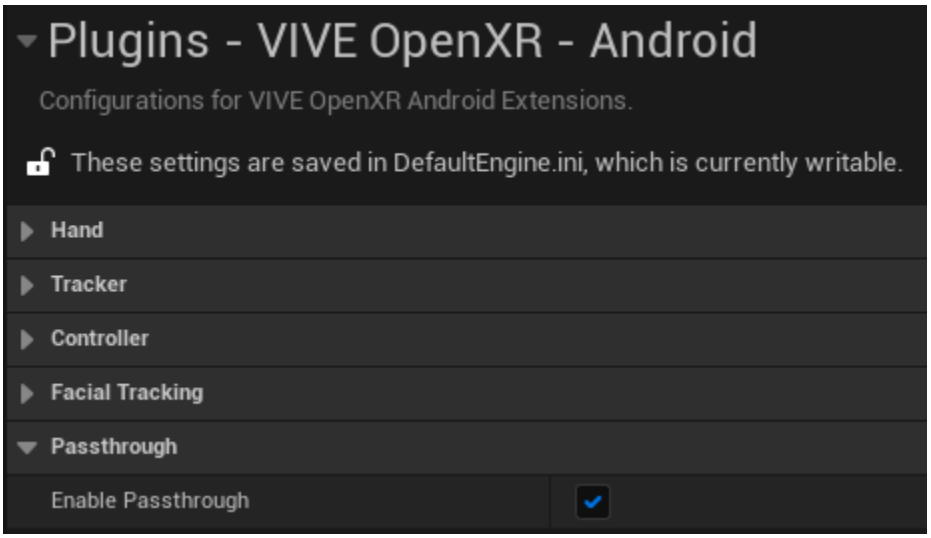
6.6.1 Introduction

Passthrough Underlay is a feature which allows you to show passthrough image under content layer.

You can refer to the official OpenXR documentation on the XR_HTC_passthrough extension for more information.

6.6.2 Project Settings

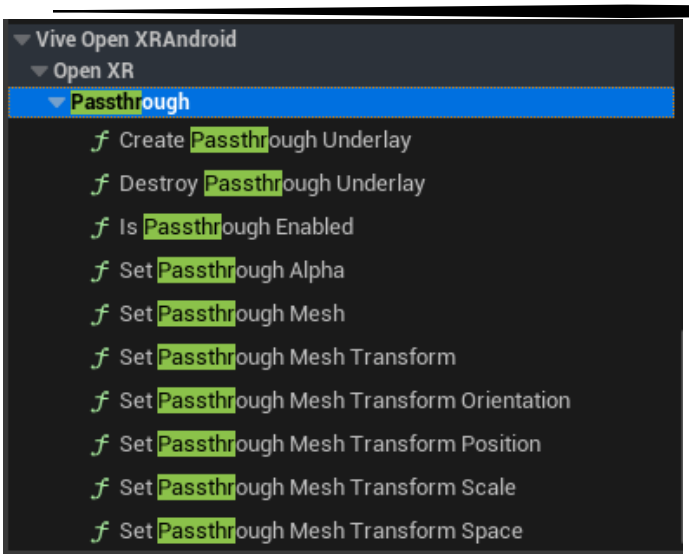
Before using **Passthrough Underlay**, you have to enable the feature in *Project Settings > Plugins – VIVE OpenXR – Android > Passthrough*.



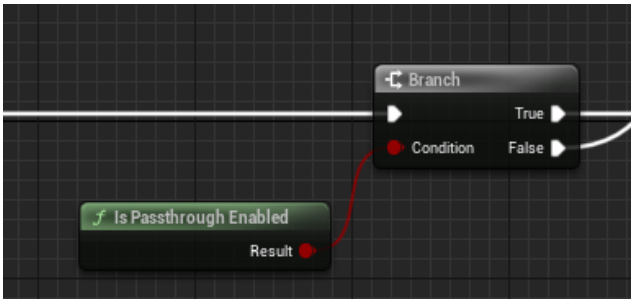
6.6.3 Blueprint Function Library

Blueprint functions are provided to you to control Passthrough Underlay.

All of the functions returns a Boolean which indicates whether the API has been successfully called or not.

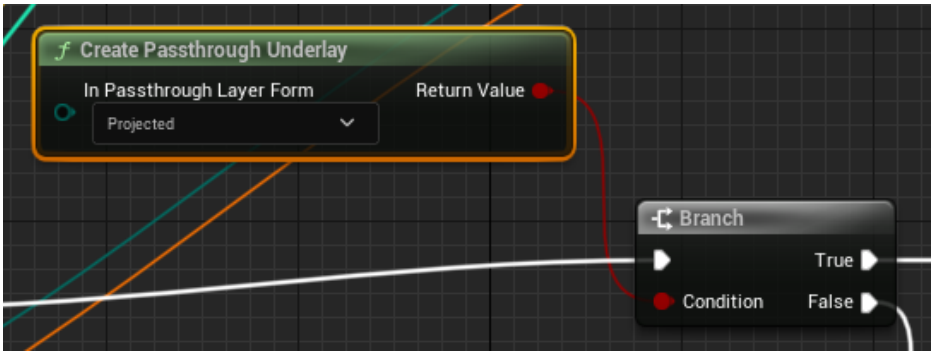


6.6.3.1 Is Passthrough Enabled



Use this function to check if the Passthrough feature is enabled or not.

6.6.3.2 Create Passthrough Underlay



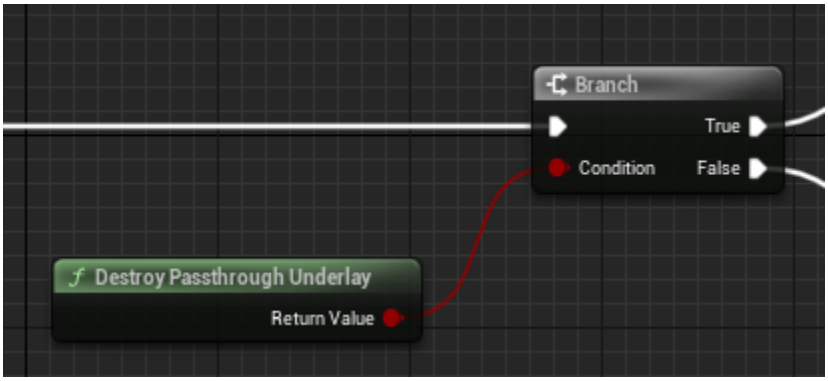
Use this function to create a Passthrough Underlay.

The “InPassthroughLayerForm” parameter specifies the type of Passthrough Underlay to be created.

Planar form is a fullscreen passthrough that covers the entire FOV.

Projected form is a passthrough that only covers part of the FOV, which the area is determined by a custom mesh and its transform.

6.6.3.3 Destroy Passthrough Underlay



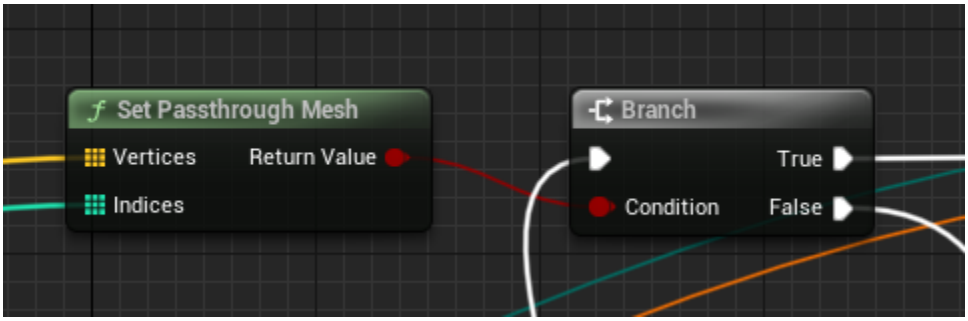
Use this function to destroy any Passthrough Underlay created previously.

6.6.3.4 Set Passthrough Alpha

Use this function to set the opacity of the Passthrough Underlay.

The “alpha” parameter specifies the target opacity (Clamped to range [0,1]).

6.6.3.5 Set Passthrough Mesh

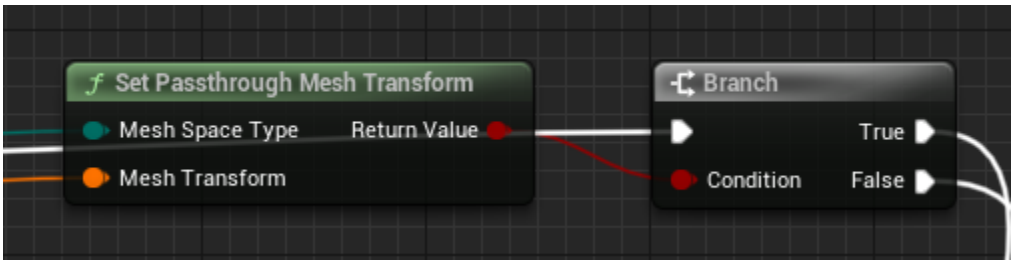


Use this function to specify the mesh data of the customized mesh for Projected Passthrough Underlay.

“vertices” specifies the vertices of the mesh.

“indices” specifies the triangle indices of the mesh.

6.6.3.6 Set Passthrough Mesh Transform

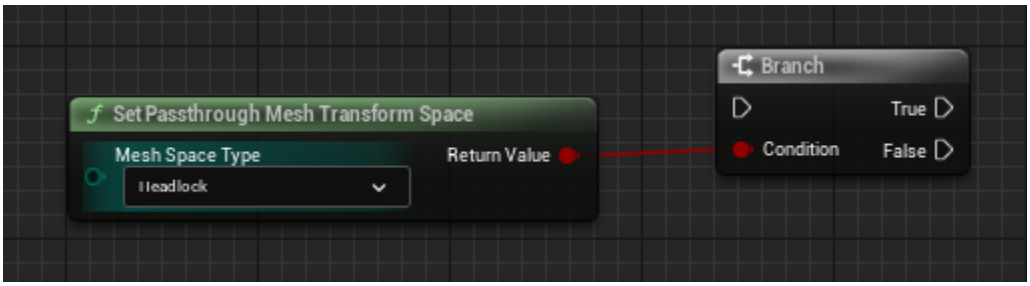


Use this function to set the space type and the transform of the mesh to be used by Projected Passthrough Underlay.

“meshSpaceType” specifies the XrSpace configuration to be used by Projected Passthrough Underlay, which is defined as follows:

```
enum class EProjectedPassthroughSpaceType : uint8 {
	//XR_REFERENCE_SPACE_TYPE_VIEW at (0,0,0) with orientation (0,0,0,1)
	Headlock = 0,
	// When TrackingOrigin is EHMDTrackingOrigin::Stage:
	// XR_REFERENCE_SPACE_TYPE_STAGE at (0,0,0) with orientation (0,0,0,1)
	//
	// When TrackingOriginMode is EHMDTrackingOrigin::Eye:
	// XR_REFERENCE_SPACE_TYPE_LOCAL at (0,0,0) with orientation (0,0,0,1)
	Worldlock = 1
};
```

6.6.3.7 Set Passthrough Mesh Transform Space



Use this function to set the space type to be used by Projected Passthrough Underlay.

Refer to 6.6.3.6 for details on “meshSpaceType”.

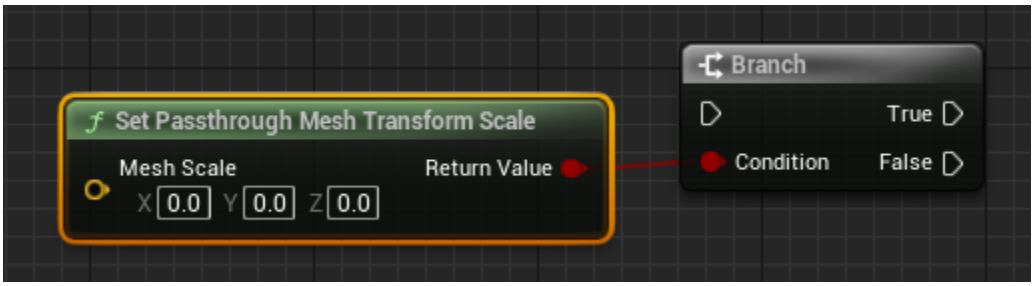
6.6.3.8 Set Passthrough Mesh Transform Position

Use this function to set the position of the Projected Passthrough Underlay.

6.6.3.9 Set Passthrough Mesh Transform Orientation

Use this function to set the orientation of the Projected Passthrough Underlay.

6.6.3.10 Set Passthrough Mesh Transform Scale



Use this function to set the scale of the Projected Passthrough Underlay.

6.6.4 Sample Blueprint

The sample “Passthrough Test” can be found at OpenXRSample/Content/PassthroughTest in the OpenXRSample project.

7 Known Issues

1. *[UE4.27] Stereo Layer might flicker after show/hide multiple times. The problem had been fixed in UE5.0. Recommend to use UE5.0 instead.*

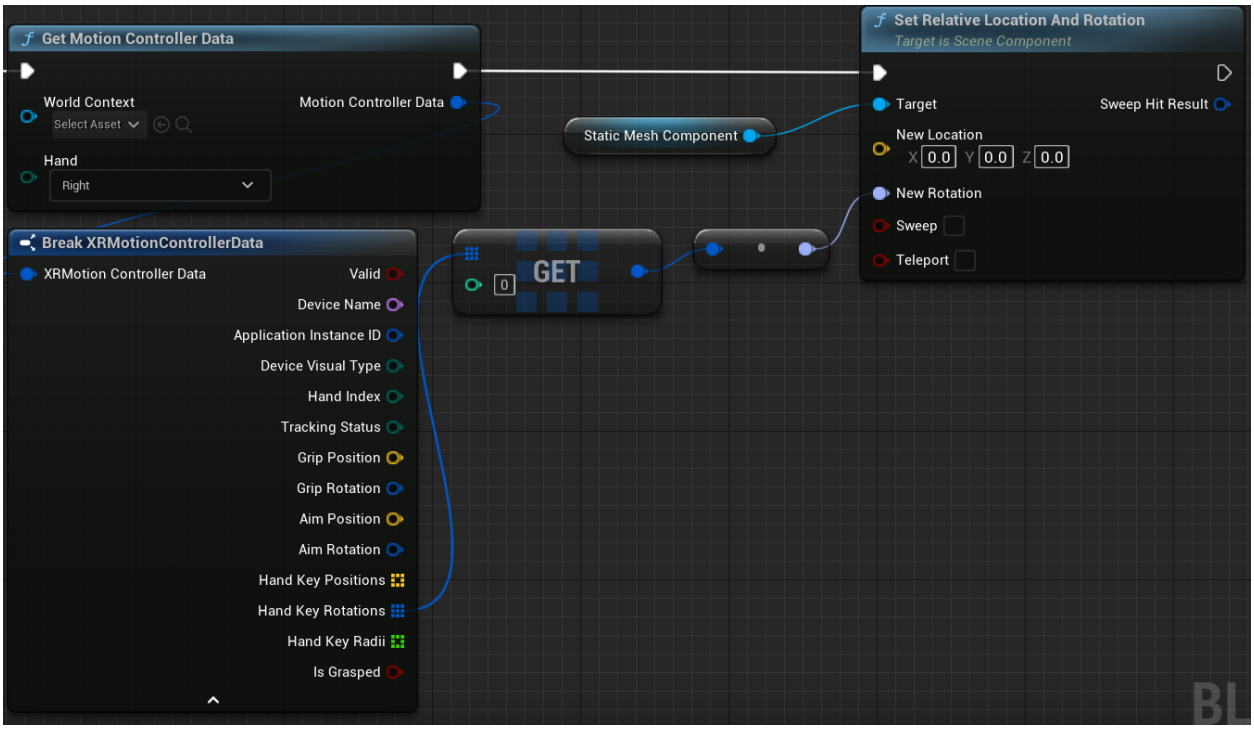
8 Troubleshooting

1. *Apps with development build may crash after turning off HMD by pressing power bank, take off for a while then put it on or switching app out and back from another one.*

[How to fix]

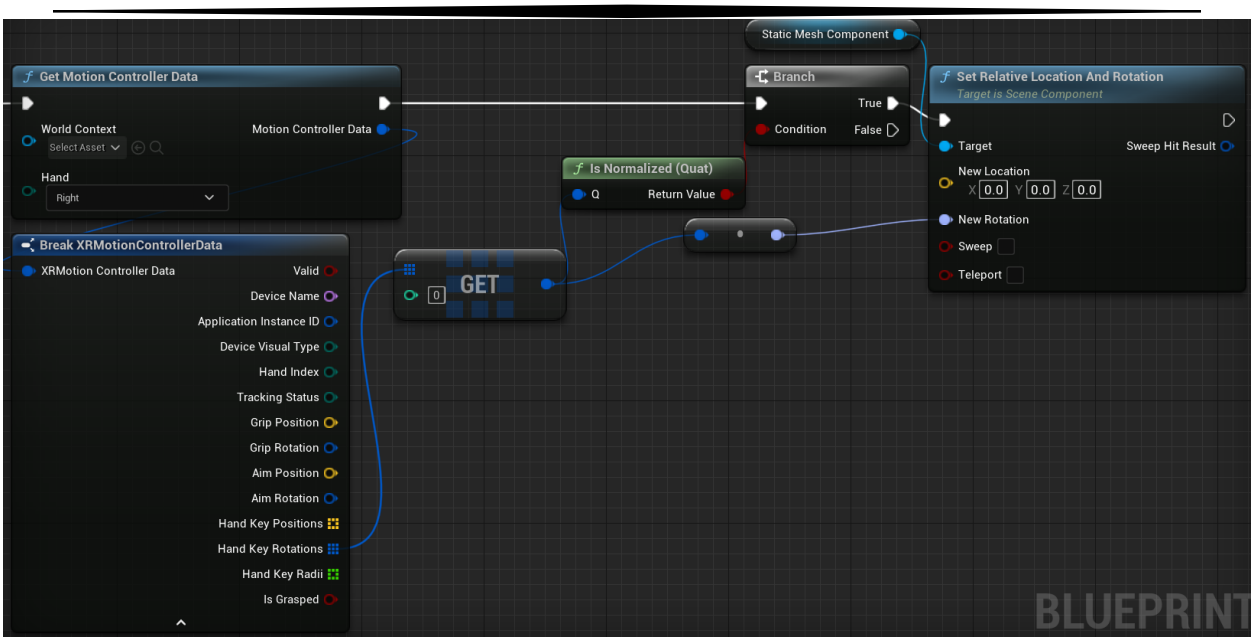
Select *project settings > Project > Packaging > Project > Build Configuration* as Shipping.

2. *[UE5][HandTracking][ShippingBuild] The Actor may disappear when you convert Hand Key Rotation (Quat) to Rotator and set the rotation to the Actor.*



[How to fix]

Make sure the Quat **IsNormalized(Quat)** first.



3. [UE5.1] Crash after hide and show StereoLayer.

[How to fix] Enable **Live Texture**.