



# Build Unreal Engine 4.27 / 5.0 OpenXR applications on VIVE Focus3

August. 2022

This document contains information that is proprietary to HTC Corporation.

Unauthorized reproduction or disclosure of this information in whole or in part is strictly prohibited.

© 2022 HTC Corporation. All rights reserved.

## Revision History

Revision	Date	Description
1.0.0	May 2022	Formal release.
1.0.1	Jul 2022	<ul style="list-style-type: none"> <li>- Supported UE5.0.</li> <li>- Added Hand Interaction and Wrist Tracker.</li> </ul>
1.0.2	Aug 2022	<ul style="list-style-type: none"> <li>- Supported UE OpenXREyeTracker plugin.</li> <li>- Added ViveFocus3 controller model in sample zip.</li> <li>- Renamed plugin folder name to ViveOpenXRAndroid.</li> </ul>

---

<b>1.</b>	<b>Introduction</b>	<b>5</b>
<b>2.</b>	<b>Download Pre-built Unreal Engine</b>	<b>6</b>
2.1	Download Epic Games Launcher.	6
2.2	Download Unreal Engine by Epic Games Launcher	6
<b>3</b>	<b>Vive Wave OpenXR - Android SDK</b>	<b>7</b>
3.1	Vive Wave OpenXR - Android Plug-In	7
3.2	How to import Vive Wave OpenXR - Android Plug-In	7
3.3	OpenXRSample	7
<b>4</b>	<b>Package the Android Sample</b>	<b>8</b>
4.1	Project Settings of Android Environments	8
4.1.1	Location of Android SDK	8
4.1.2	Location of Android NDK	9
4.1.3	Location of JAVA	10
4.2	Package the ButtonTest to an Android APK	11
<b>5</b>	<b>Porting scene to use Vive Wave OpenXR - Android</b>	<b>12</b>
5.1	Import Vive Wave OpenXR Android Plug-In	13
5.2	Disable other Virtual Reality Plugin-Ins	13
5.3	Project Settings	13
5.3.1	Project Settings of Unreal Engine	13
5.3.2	Project Settings of VIVE OpenXR Android Plug-In	14
5.3.2.1	Enable Hand Interaction	15
5.3.2.2	Enable Wrist Tracker	15
5.3.2.3	Enable Focus3 Controller	15
5.4	Input	15
5.5	Vive Focus 3 Controller Model	16
5.5.1	Download the sample zip	16
5.5.2	Setup the Controller Model in MotionController	17
<b>6</b>	<b>Advanced Features</b>	<b>18</b>
6.1	Hand Tracking	18
6.2	Hand Interaction	18
6.2.1	Project Setting	18
6.2.2	Blueprint Function Libraries	19
6.2.2.1	Get Pinch Active	19
6.2.2.2	Get Pinch Tracked	19
6.2.2.3	Get Pinch Position	19
6.2.2.4	Get Pinch Rotation	20
6.2.2.5	Get Pinch Strength	20
6.2.3	Wave Hand Pointer	20
6.3	Wrist Tracker	21
6.3.1	Project Settings	22
6.3.2	Blueprint Function Libraries	22
6.3.2.1	Get Tracker Active	22

---

---

6.3.2.2	Get Tracker Tracked	23
6.3.2.3	Get Tracker Position	23
6.3.2.4	Get Tracker Rotation	23
6.3.2.5	Get Tracker Key Down	23
6.3.3	Key event of Tracker input	24
6.4	Eye Tracking	26
6.4.1	Eye Gaze	26
<b>7</b>	<b>Known Issues</b>	<b>27</b>
<b>8</b>	<b>Troubleshooting</b>	<b>28</b>

# 1. Introduction

VIVE Wave runtime had already supported OpenXR standard on VIVE Focus3. Welcome to join OpenXR of Wave runtime.

Since Unreal Engine 4.27, Epic Games claimed that the OpenXR Plug-in which allowed to develop the cross-platform applications for OpenXR is Production-Ready. In this guide, you will learn how to build an OpenXR app running on Focus3 through the following step-by-step sections.

What you will learn in this guide:

- Download pre-built Unreal Engine 4.27 / 5.0 through Epic Games Launcher
- Package, install and run Vive Wave OpenXR - Android application on VIVE Focus3
- Porting scene to use Vive Wave OpenXR - Android SDK

What Wave support on Focus3 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

Okay. Let's get started to create your Vive Wave OpenXR - Android content on Focus3!

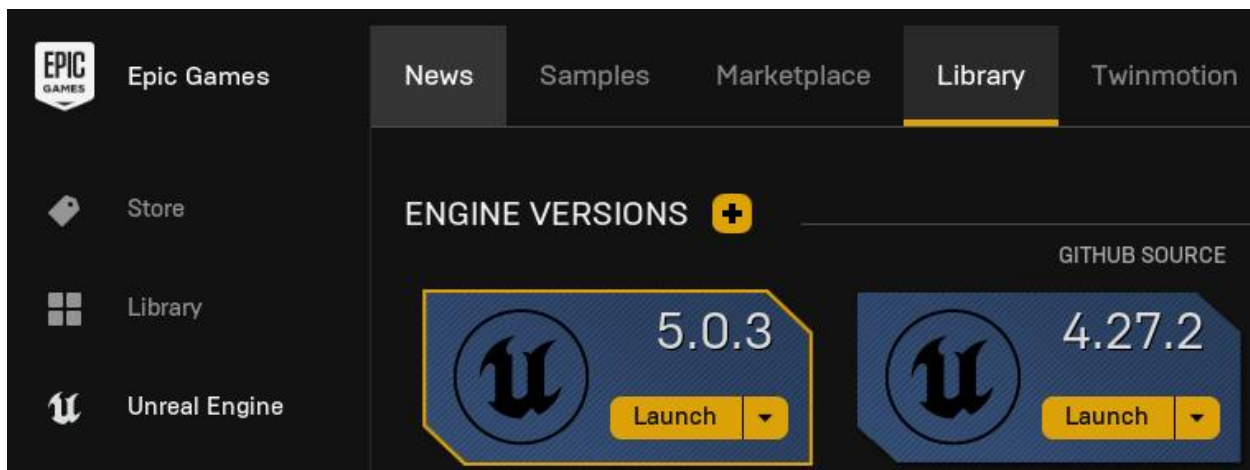
## 2. Download Pre-built Unreal Engine

### 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 **4.27.2** or **5.0.3**.



---

## 3 Vive Wave OpenXR - Android SDK

### 3.1 Vive Wave OpenXR - Android Plug-In

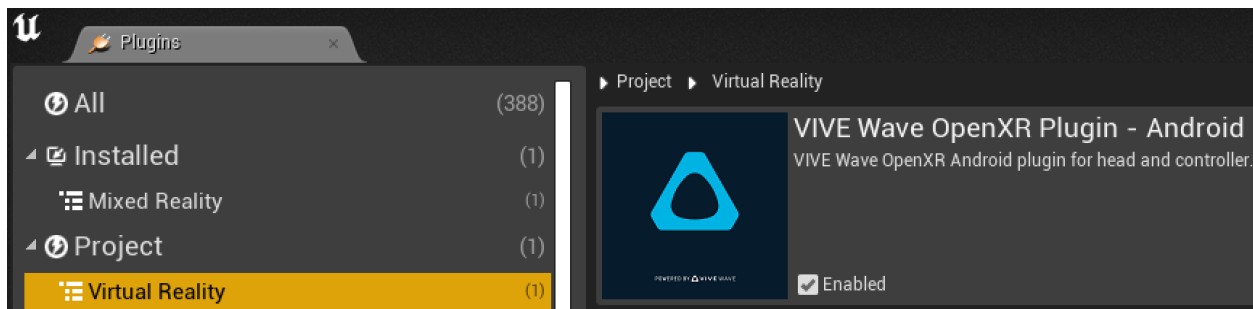
The **Vive Wave OpenXR - Android Plug-In** is at `<Where you unzip wave_openxr_plugin_<version>.zip>\ViveOpenXRAndroid`.

### 3.2 How to import Vive Wave 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. Double-click `<Your project>.uproject` to launch the project.

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



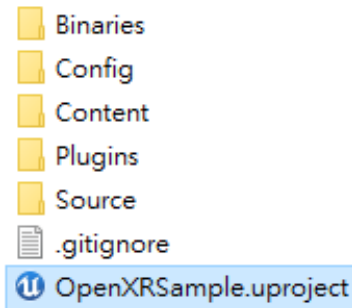
### 3.3 OpenXRSample

OpenXRSample is a sample project which includes 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 Wave OpenXR - Android Plug-In before packaging the android apps.

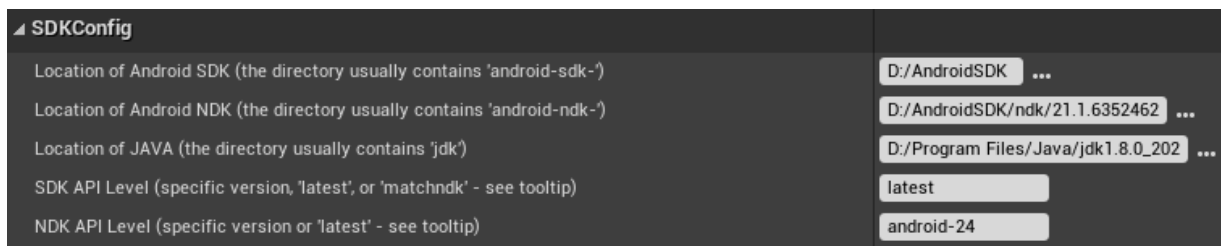
## 4 Package the Android Sample

Please make sure you already import the Vive Wave 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**.

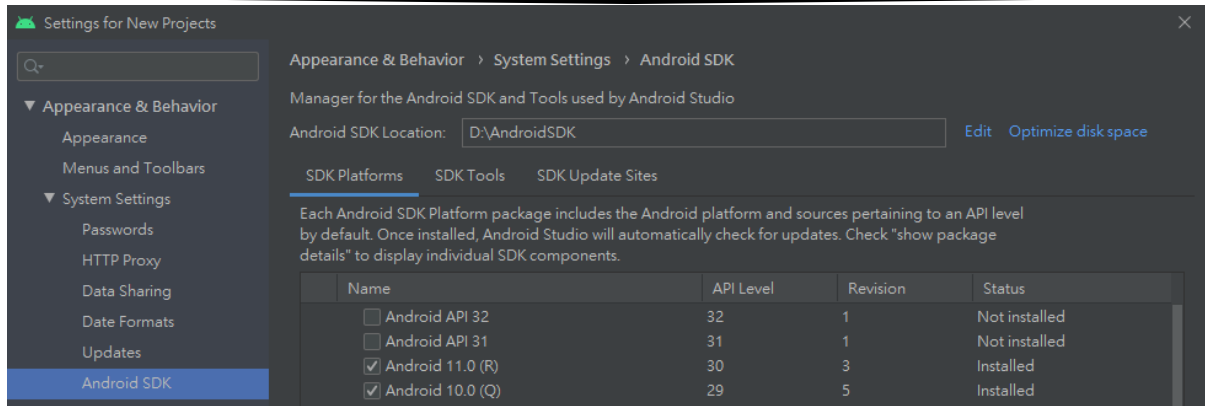


You could refer [Setting up Android SDK and NDK for Unreal](#) to set up.

#### 4.1.1 Location of Android SDK

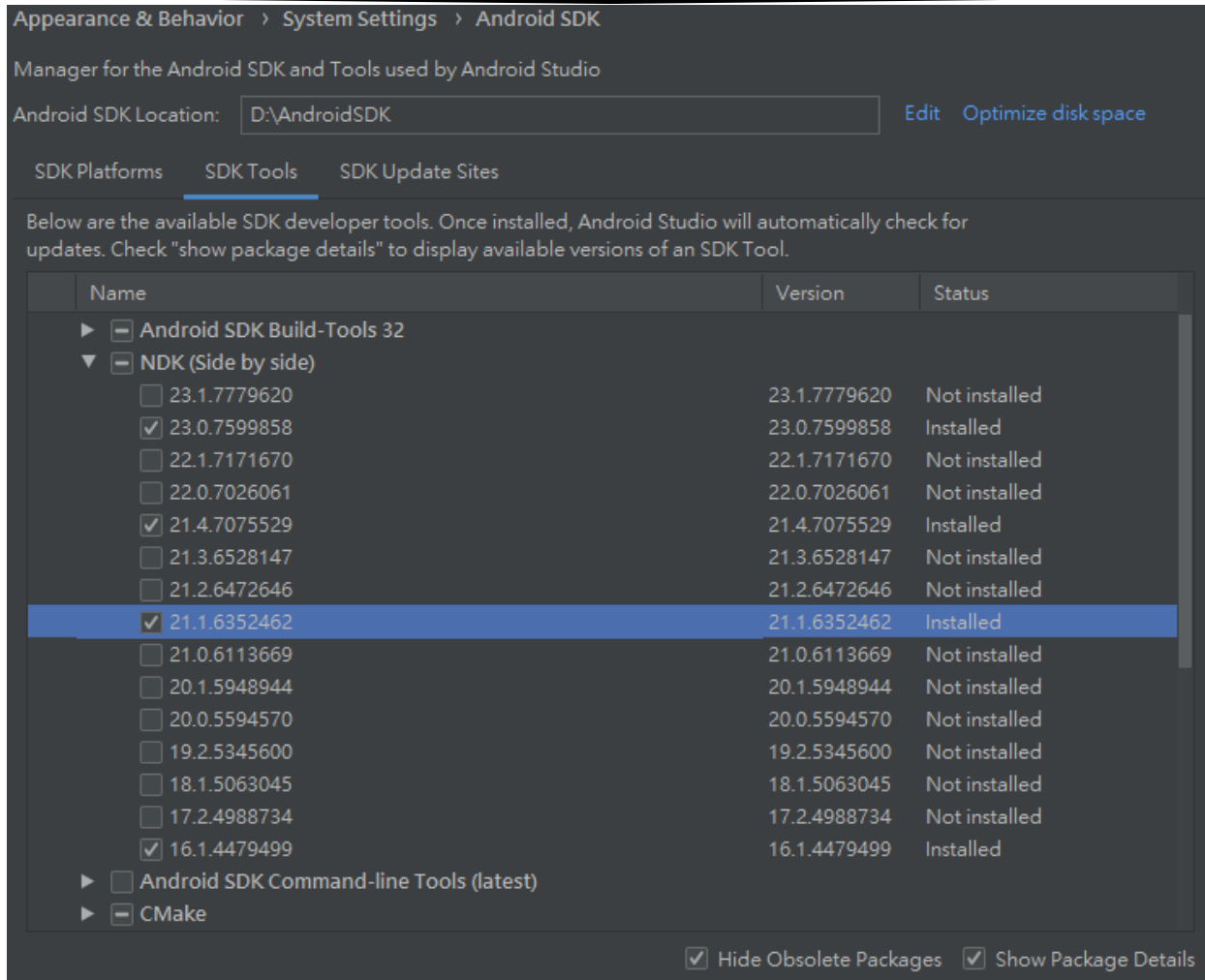
Download Android Studio if you didn't have yet. In Android Studio Editor, click *Tools > SDK Manager* and navigate to *Appearance & Behavior > System Settings > Android SDK*. The **Android SDK Location** is the path to be filled.





## 4.1.2 Location of Android NDK

Please fill your Android NDK(**21.1.6352462**) path to it. 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.

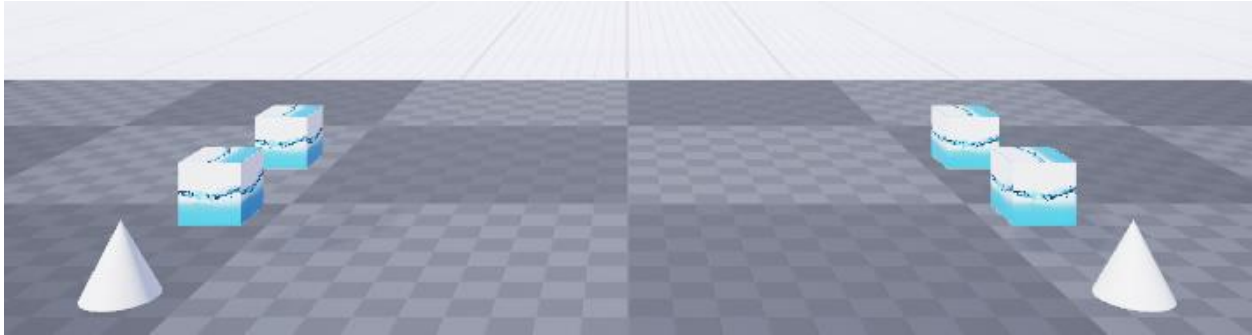


### 4.1.3 Location of JAVA

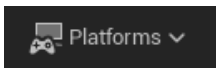
It is recommended to install [jdk1.8.0](#).

## 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*.



In UE4.27, click **File > Package Project > Android > Android(ASTC)** to package the android application. In UE5.0, click **icon Platforms > Android > Package Project** to package it.



Then you will see the `OpenXRSample-arm64.apk` in the path *<path to OpenXRSample>/Android\_ASTC/*.

Then you can execute **Install\_OpenXRSample-arm64.bat** to install the android application automatically. For more information, refer to [Android Getting Started](#).

Launch the sample **OpenXRSample** in Focus3, you can use right controller to control right objects and left controller to control left objects.

The far cube presents the button “touch” and “axis” actions. The far cube will change color when touching a button and rotate when the button has axis. Only **Trigger** and **Grip** button have axis.

The near cube presents the button “press” action and will change color when pressing a button.

The cone presents the **Thumbstick** axis and will move when you operate the **Thumbstick** button.

## 5 Porting scene to use Vive Wave OpenXR - Android

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

In UE4.27, select Virtual Reality project in the template list.



In UE5.0, click **GAMES** then select Virtual Reality project.



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



## 5.1 Import Vive Wave OpenXR Android Plug-In

Please refer chapter [3.2](#) above to see how to import the Vive Wave 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 Wave 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 arm64** and disable **Support armv7**. (Skip this in UE5 because UE5 did not support armv7 anymore.)



Enable **Support Vulkan**. (MUST)

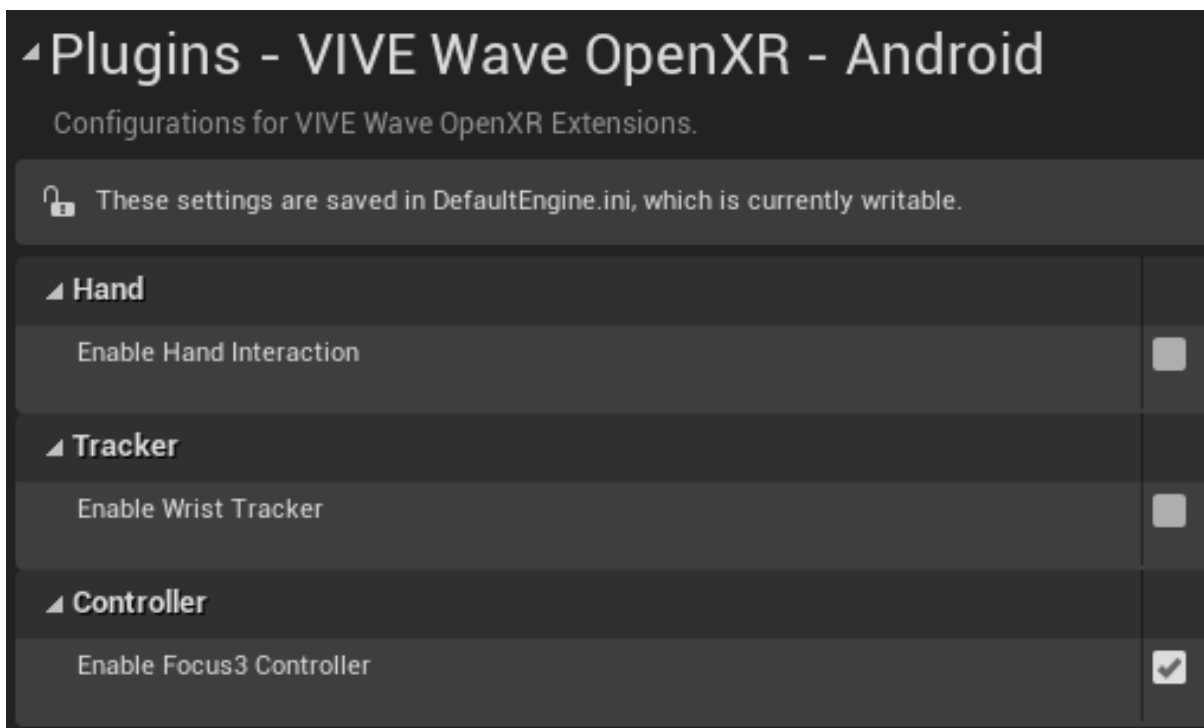


Disable **Show launch image** to get rid of the non-stereo splash while launching app.



### 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 Wave 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](#).

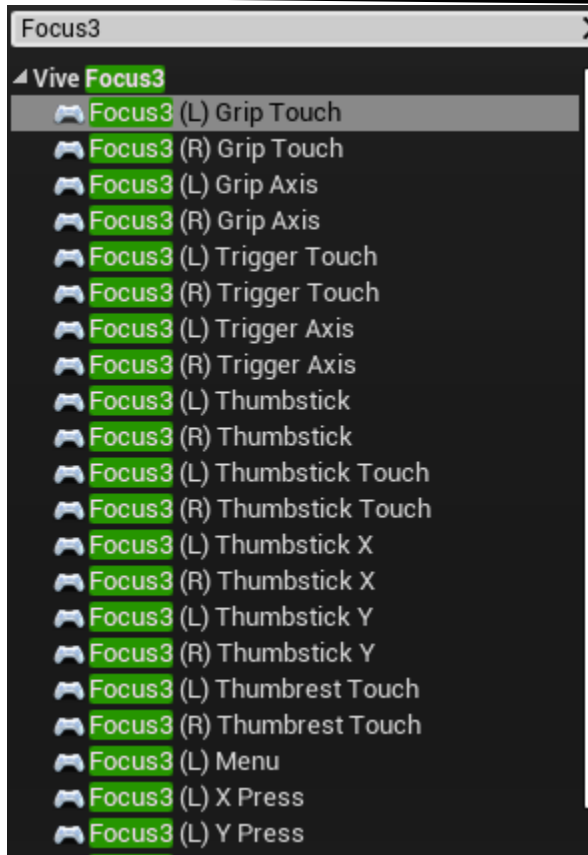
### 5.3.2.3 Enable Focus3 Controller

Select this option to support Focus3 controller.

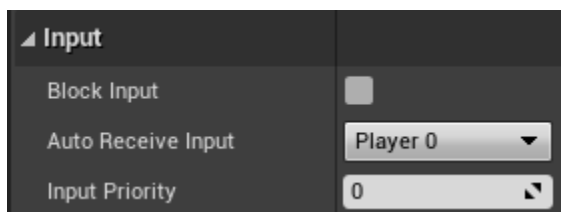
## 5.4 Input

To use the Focus3 controller input, you have to apply the **Input** action and axis mappings. Refer to this [guide](#) for detailed information.

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.



## 5.5 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.

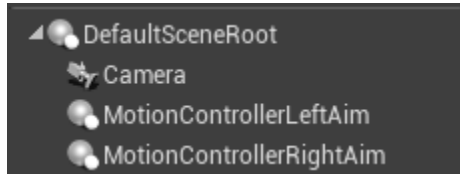
### 5.5.1 Download the sample zip

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



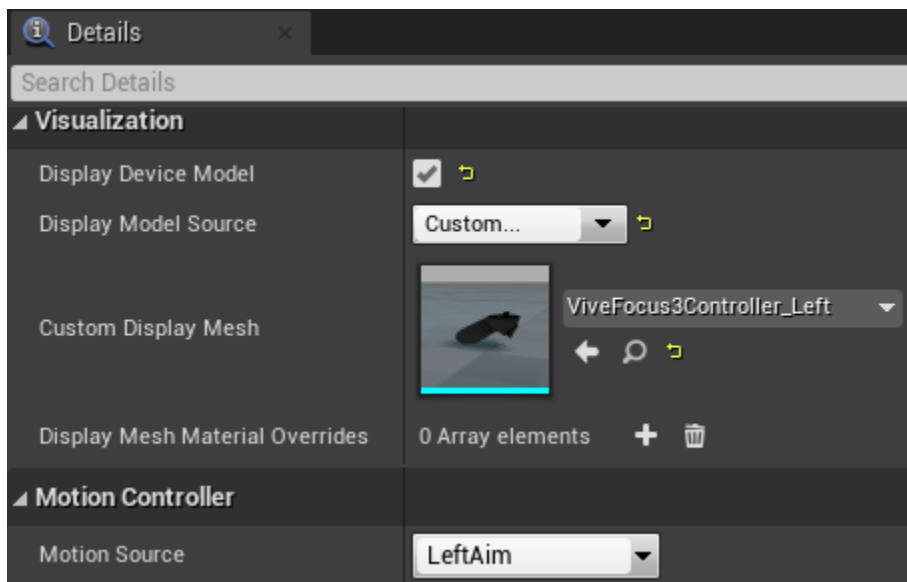
## 5.5.2 Setup the Controller Model in MotionController

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



Click the MotionController component and check these settings:

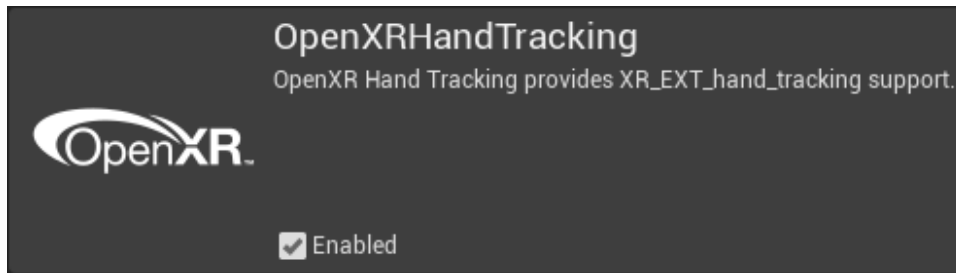
1. Enable **Display Device Model**.
2. Set **Display Model Source** as Custom.
3. Select **Custom Display Mesh** as ViveFocus3Controller with the desired hand.
4. Select the **Motion Source** as LeftAim or RightAim.



---

## 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

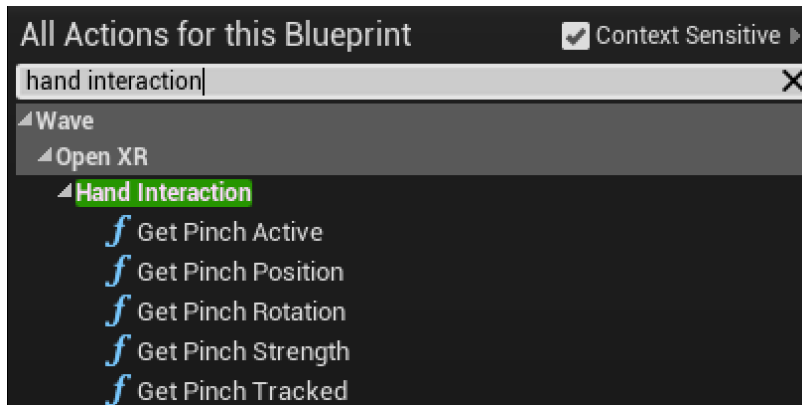
Use pinch motion to interact with Actors.

#### 6.2.1 Project Setting



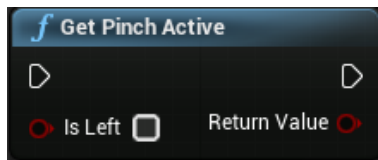
Click **Edit > Project Settings** to open the Project Settings window and navigate to **Plugins > VIVE Wave OpenXR - Android**. Select **Enable Hand Interaction** to enable this feature.

## 6.2.2 Blueprint Function Libraries



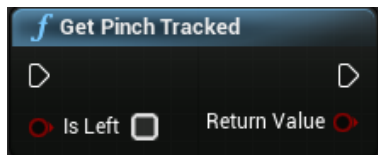
You can retrieve these pinch status of left hand if **Is Left** is true and the status of right hand if **Is Left** is false.

### 6.2.2.1 Get Pinch Active



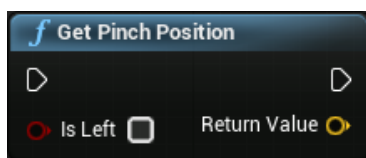
Checks if the pinch motion is active.

### 6.2.2.2 Get Pinch Tracked



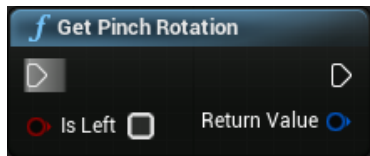
Checks if the pinch motion is tracked.

### 6.2.2.3 Get Pinch Position



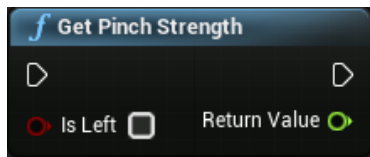
Retrieves the position of pinch origin of left or right hand.

### 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



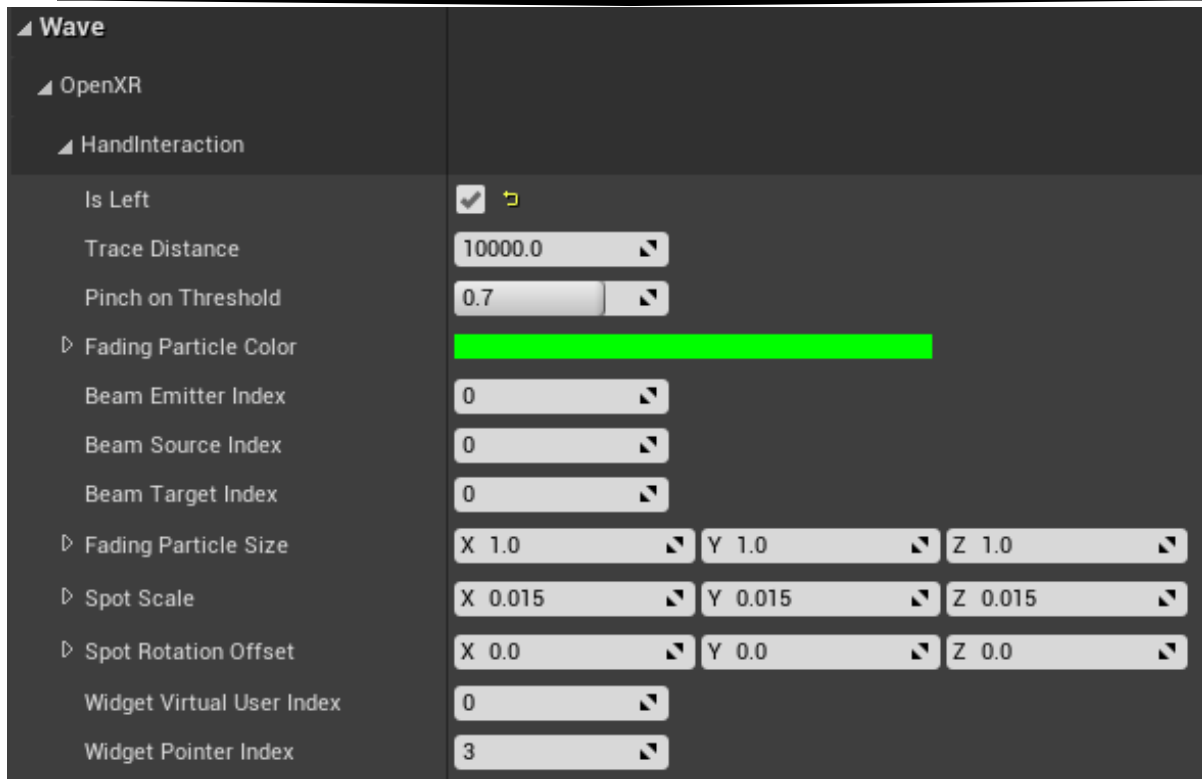
Retrieves the pinch strength of left or right hand. The return value should be [0, 1]. 1 means the index finger is closed to thumb very much.

### 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.

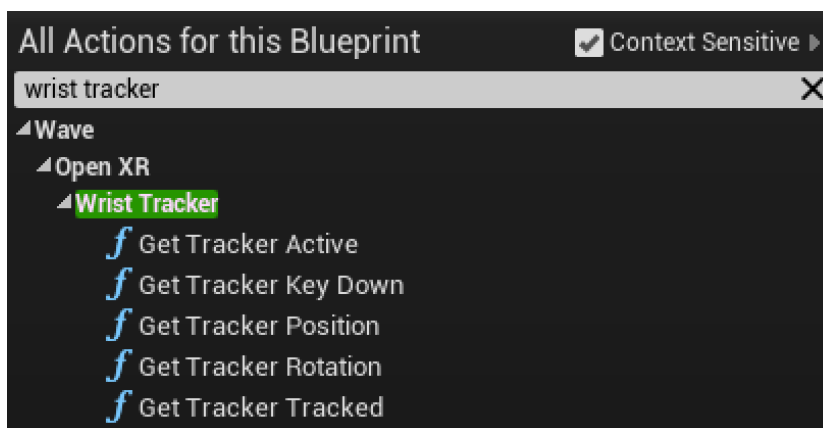


### 6.3.1 Project Settings



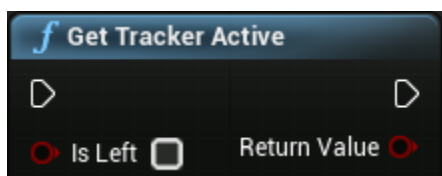
Click **Edit > Project Settings** to open the Project Settings window and navigate to **Plugins > VIVE Wave OpenXR - Android**. Select **Enable Wrist Tracker** to enable this feature.

### 6.3.2 Blueprint Function Libraries



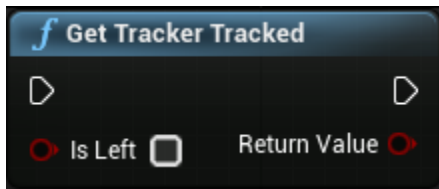
You can retrieve these tracker status of left or right device by setting **Is Left**. The **L** / **R** light mark the device is left or right. The **L** light will be on if the tracker is the left device.

#### 6.3.2.1 Get Tracker Active



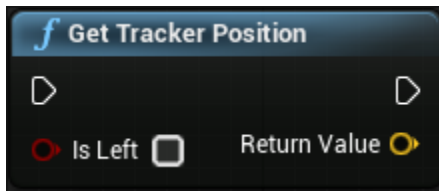
Checks if the tracker is active. (The device is connected or not.)

### 6.3.2.2 Get Tracker Tracked



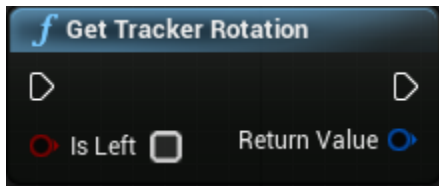
Checks if the tracker is tracked.

### 6.3.2.3 Get Tracker Position



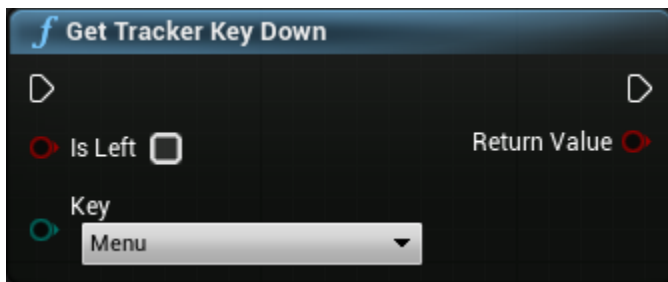
Retrieves the position of the left or right tracker.

### 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.



## Engine - Input

Input settings, including default input action and axis bindings.

🔒 These settings are saved in DefaultInput.ini, which is currently writable.

### Bindings

Action and Axis Mappings provide a mechanism to conveniently map keys and axes to input behaviors. Mappings allow for inputs that have a continuous range.

#### Action Mappings + 🗑️

- GrabLeft + ✕
- GrabRight + ✕
- TriggerLeft + ✕
- TriggerRight + ✕
  - Oculus Touch (R) Trigger Shift  Ctrl  Alt  Cmd  ✕
  - Valve Index (R) Trigger Shift  Ctrl  Alt  Cmd  ✕
  - Mixed Reality (R) Trigger Shift  Ctrl  Alt  Cmd  ✕
  - Vive (R) Trigger Shift  Ctrl  Alt  Cmd  ✕
  - Focus3 (R) Trigger Touch Shift  Ctrl  Alt  Cmd  ✕
  - Tracker (R) A Press Shift  Ctrl  Alt  Cmd  ✕

Search

- Gamepad
- Keyboard
- Mouse
- Vive Focus3
- Vive Focus3 Tracker
  - Tracker (L) Menu
  - Tracker (L) X Press
  - Tracker (R) System
  - Tracker (R) A Press
- Motion

InputAction TriggerRight

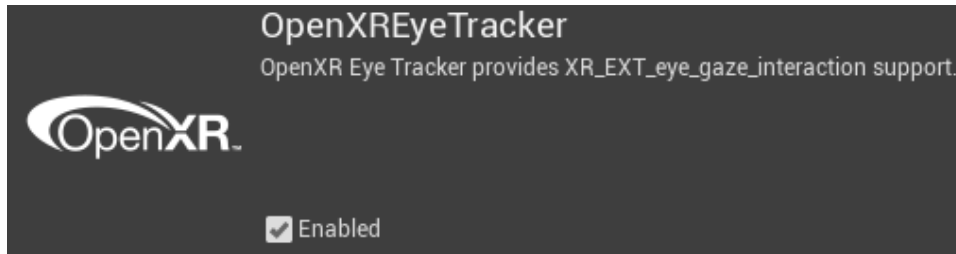
- Pressed ▶
- Released ▶
- Key 🔍

For more information, refer to [InteractiveExperiences-Input](#).

## 6.4 Eye Tracking

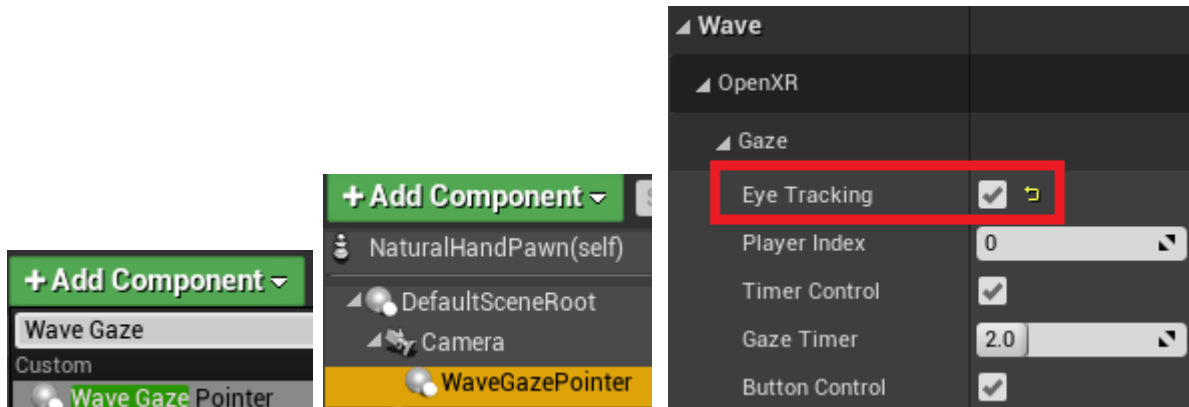
Unreal engine provides the **Eye Tracking** feature by enabling the **OpenXREyeTracker** plugin from *Plugin > Built-In > Virtual Reality > OpenXREyeTracker*.

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.

---

## 7 Known Issues

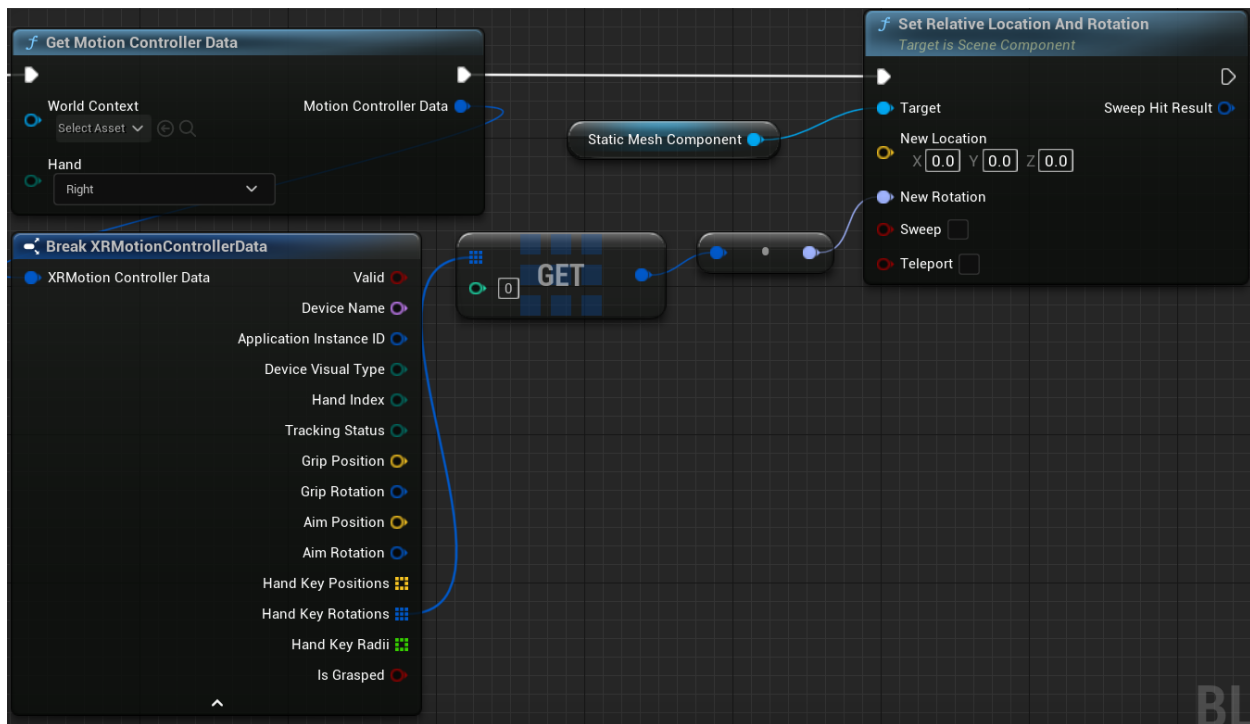
## 8 Troubleshooting

1. *Apps with development build may crash after turning off HMD by pressing power bank or take off for a while and put it on.*

[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.

