VIVE Focus

Fundamental/Interaction UNITY SDK

VIVE Focus

Contents

Introduction

The following document will explain how to set up the basic SDK's required to develop for your Pico and Hi5 V2 devices. It is recommended to use the Unity 2019.x/2020.x/2021.x LTS version to create a new project. Currently, only Pico Neo3 and Pico4 are supported, and Pico Neo2 is not supported. The Unity 2022 version is being adapted.

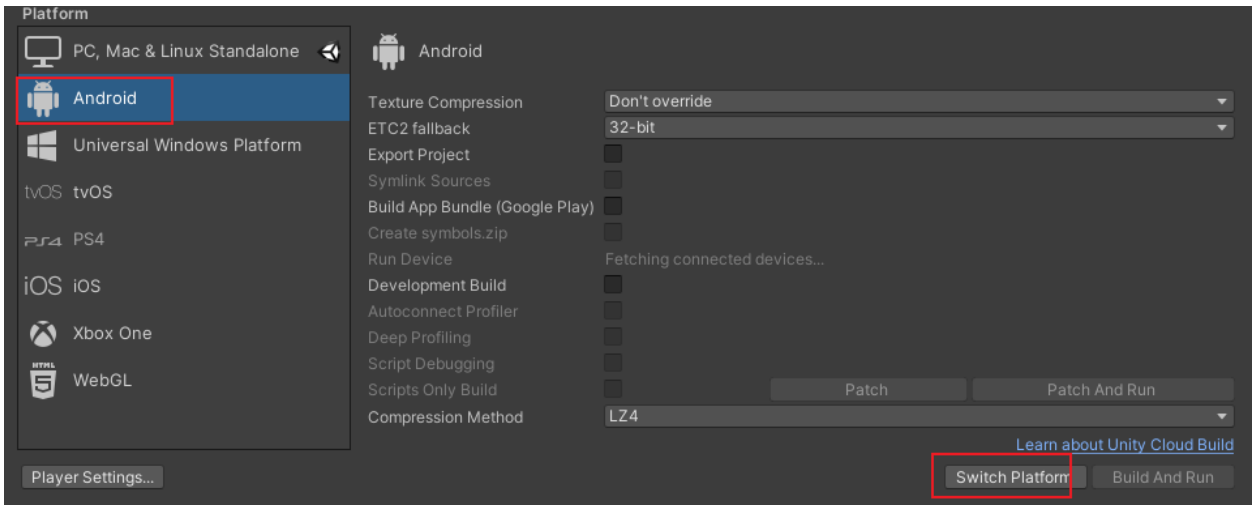
Unity Setup

Please use Unity 2019.x/2020.x/2021.x LTS version to create a new project.

Unity 2022 version is being adapted

Environment Setup

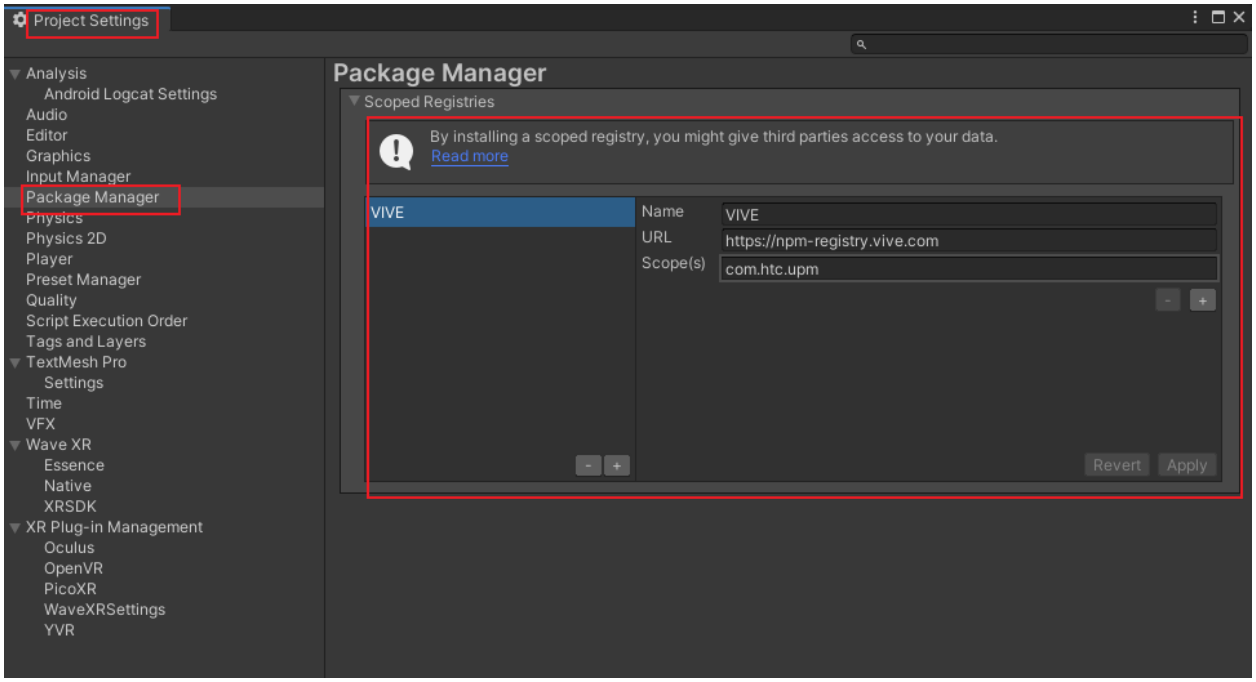
File->Build Setting->Android->Switch Platform



Setup PackageManager Edit>Project Settings>Pakcage manager

Visit the following page for details:

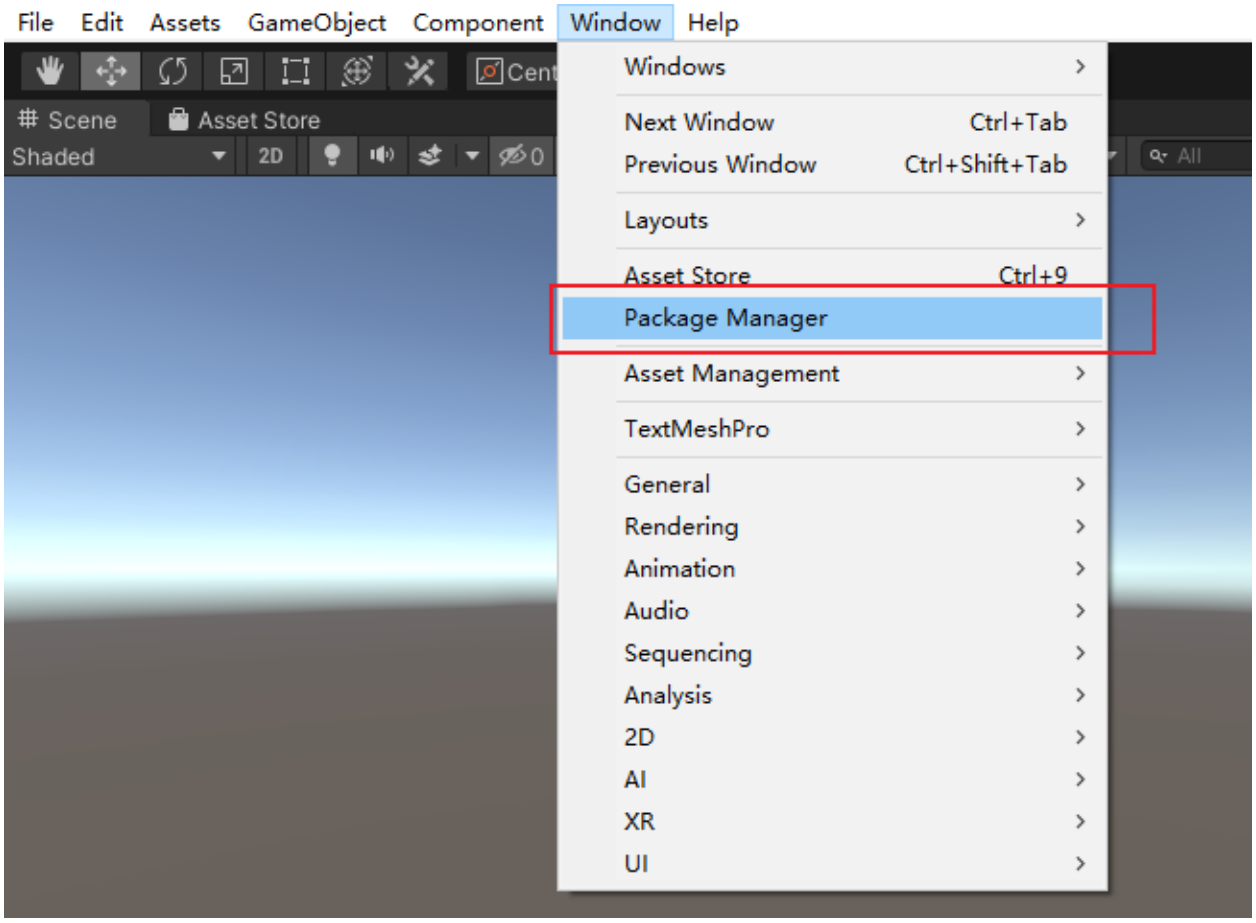
https://developer.vive.com/resources/vive-wave/tutorials/vive-registry/

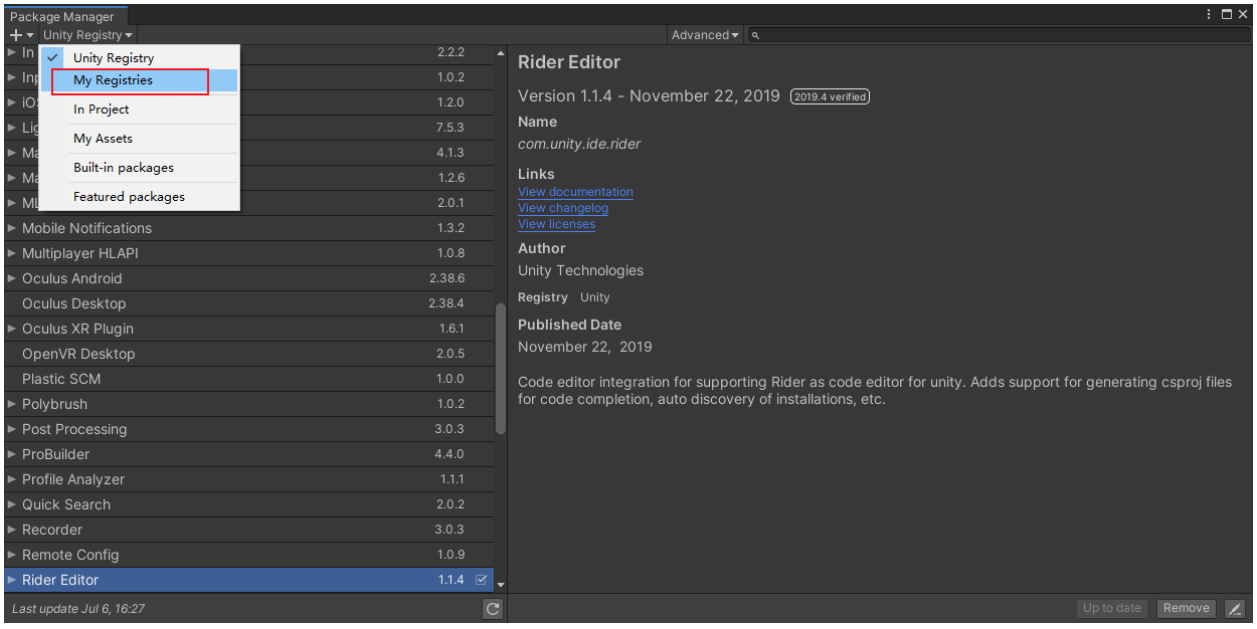


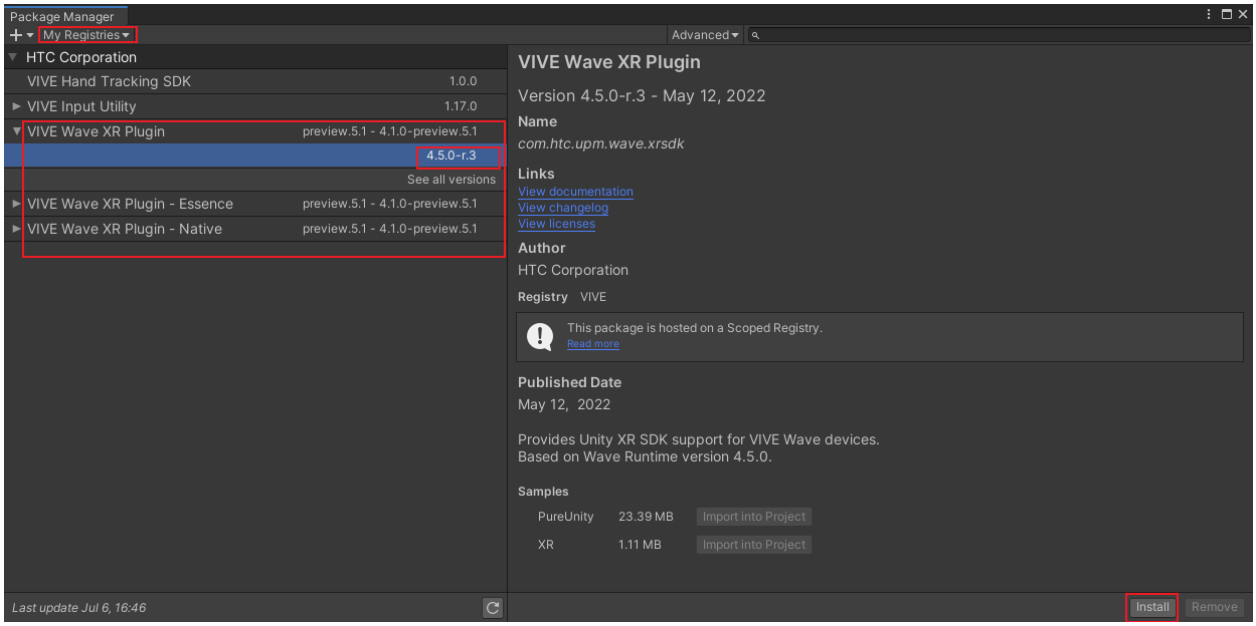
- Name: VIVE
- URL: https://npm-registry.vive.com
- Scope(s): com.htc.upm

Import the Vive-Wave Package

https://developer.vive.com/resources/vive-wave/tutorials/installing-wave-xr-plugin-unity/

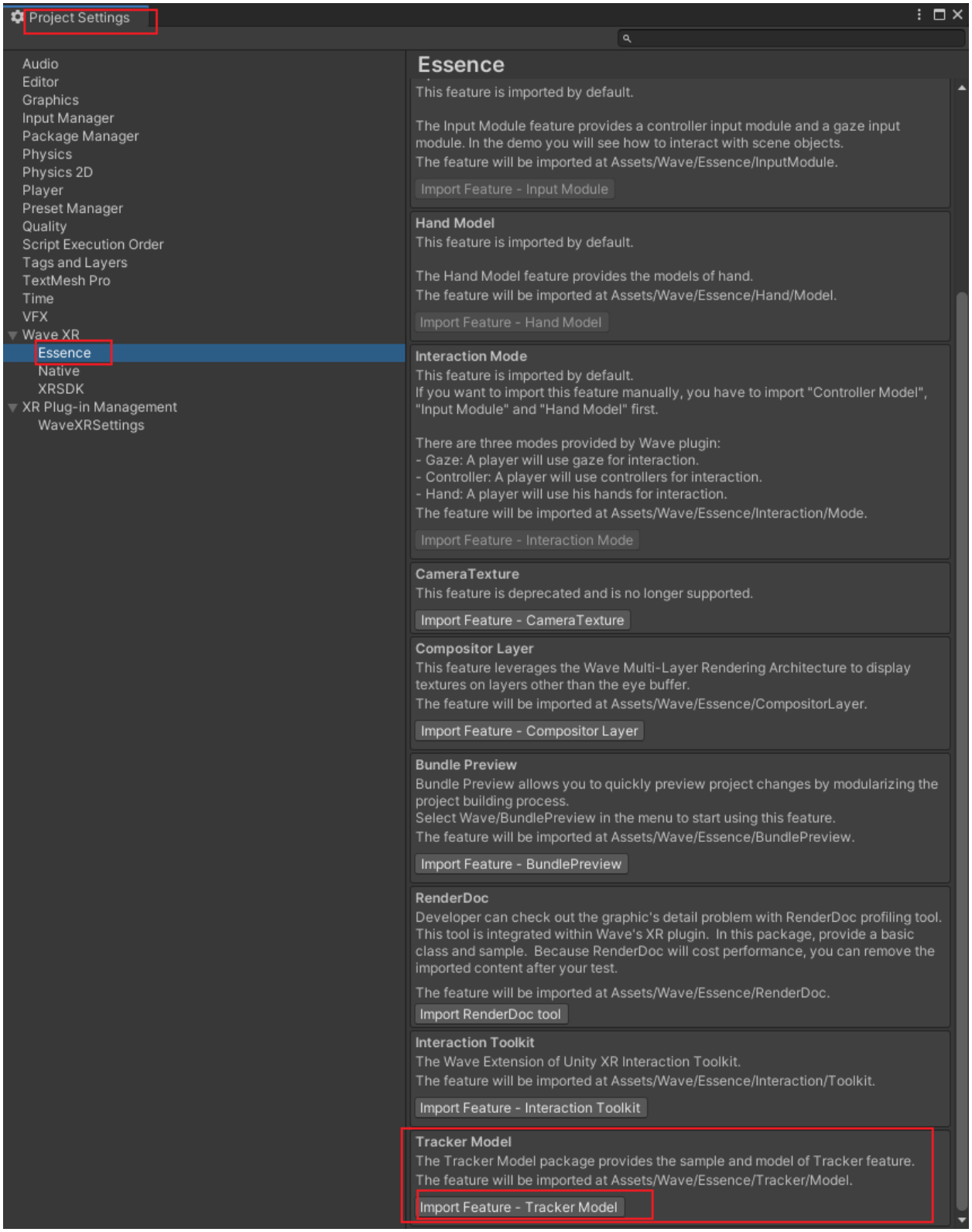

File
Edit
Assets
GameObject
Component
Window
Help
Scene
Asset Store
Shaded
2D
Windows
Next Window Ctrl+Tab
Previous Window Ctrl+Shift+Tab
Layouts
Asset Store Ctrl+9
Package Manager
Asset Management
TextMeshPro
General
Rendering
Animation
Audio
Sequencing
Analysis
2D
AI
XR
UI


Package Manager
Unity Registry
Advanced
Unity Registry
My Registries
In Project
My Assets
Built-in packages
Featured packages
Mobile Notifications 1.3.2
Multiplayer HLAPI 1.0.8
Oculus Android 2.38.6
Oculus Desktop 2.38.4
Oculus XR Plugin 1.6.1
OpenVR Desktop 2.0.5
Plastic SCM 1.0.0
Polybrush 1.0.2
Post Processing 3.0.3
ProBuilder 4.4.0
Profile Analyzer 1.1.1
Quick Search 2.0.2
Recorder 3.0.3
Remote Config 1.0.9
Rider Editor 1.1.4
Last update Jul 6, 16:27
Rider Editor
Version 1.1.4 - November 22, 2019
2019.4 verified
Name
com.unity.ide.rider
Links
View documentation
View changelog
View licenses
Author
Unity Technologies
Registry Unity
Published Date
November 22, 2019
Code editor integration for supporting Rider as code editor for unity. Adds support for generating csproj files for code completion, auto discovery of installations, etc.
Up to date
Remove

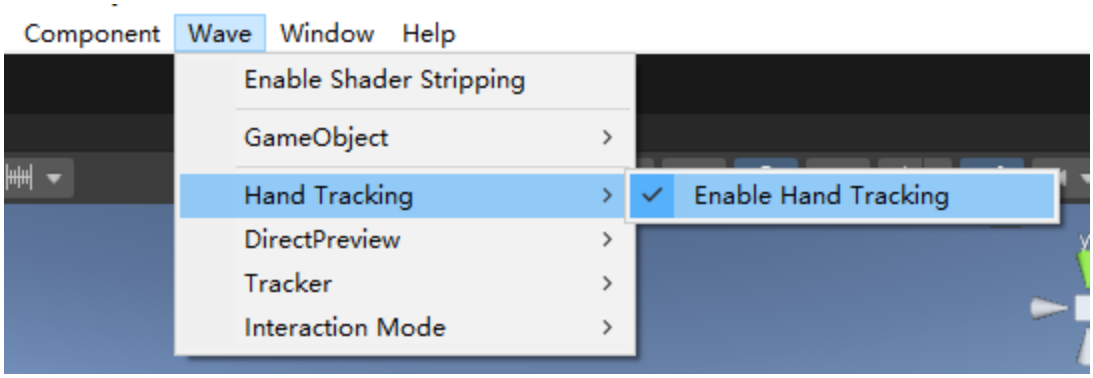


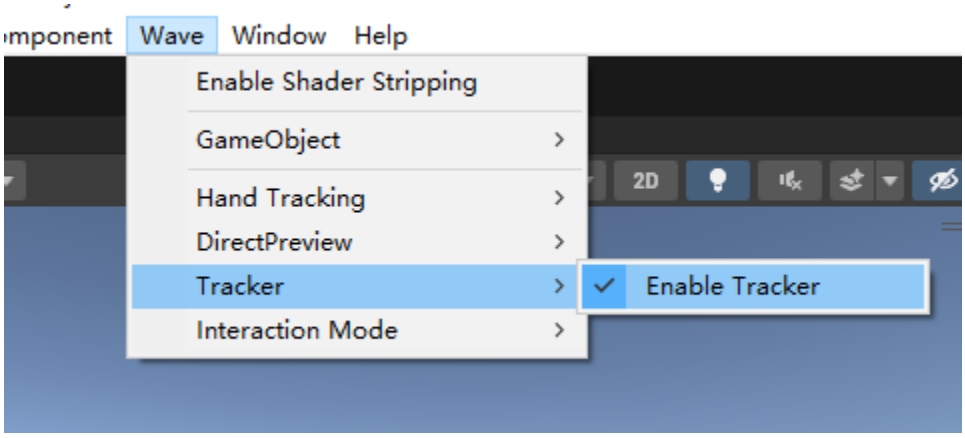
Illustrate:

1. The current version is 4.5.0-r.3
2. If you only need to use the handle, you only need to import VIVE Wave XR Plugin. If you need to develop HTC Vive Focus Wrist, you need to import all three and need to import additionally according to the following figure

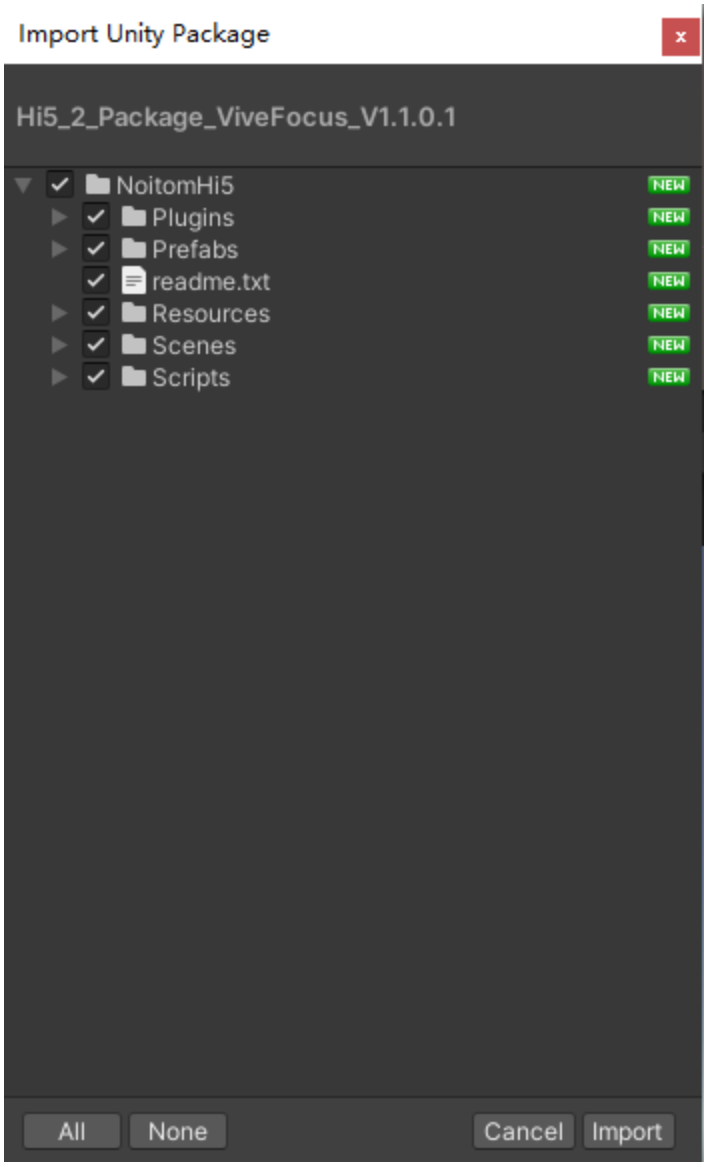


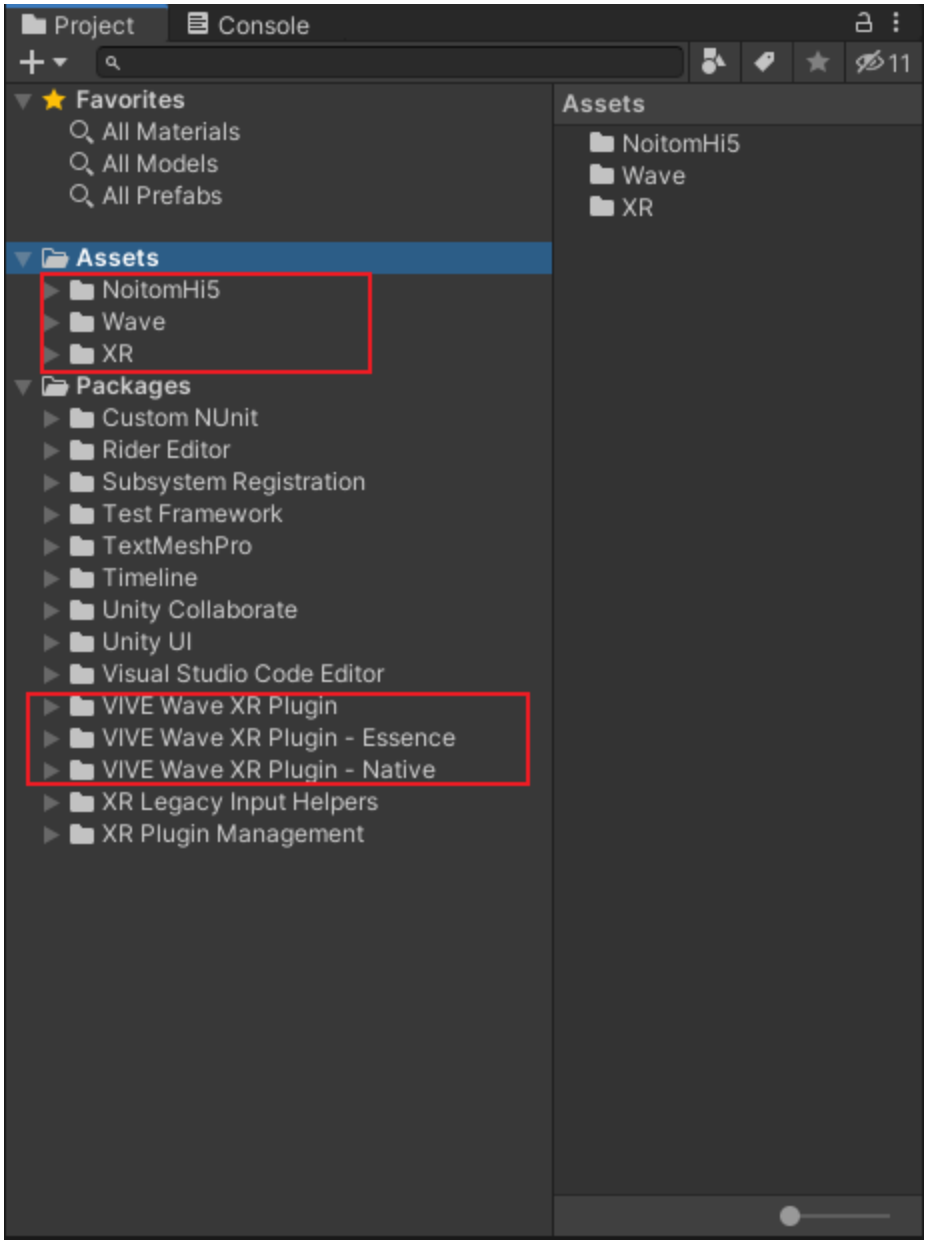
Modify the location and tracking of the wave bracelet



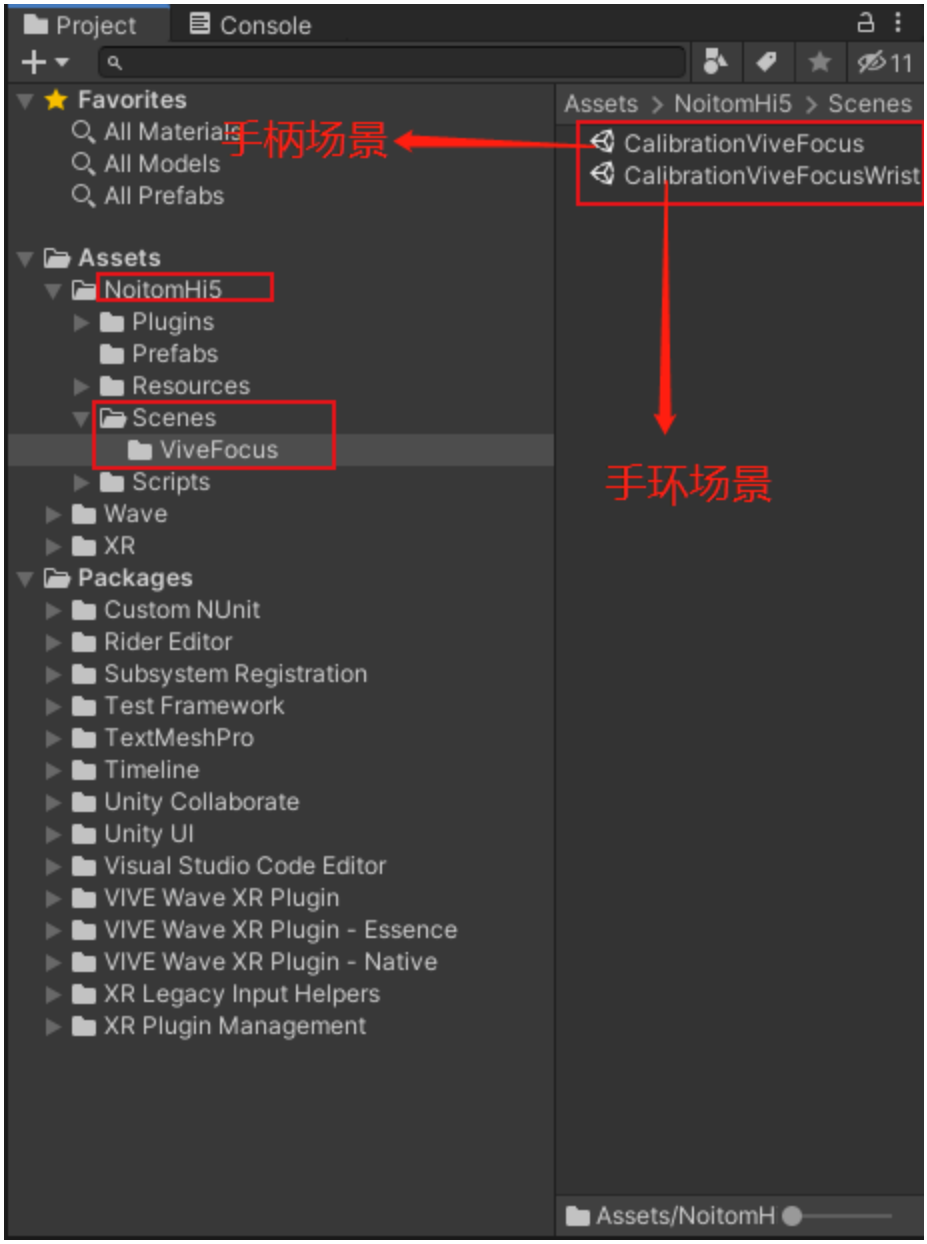


Import Hi5_2_Package_ViveFocus_V1.1.0.1.unitypackage


Import Unity Package
Hi5_2_Package_ViveFocus_V1.1.0.1
NoitomHi5
Plugins
Prefabs
readme.txt
Resources
Scenes
Scripts
NEW
All
None
Cancel
Import

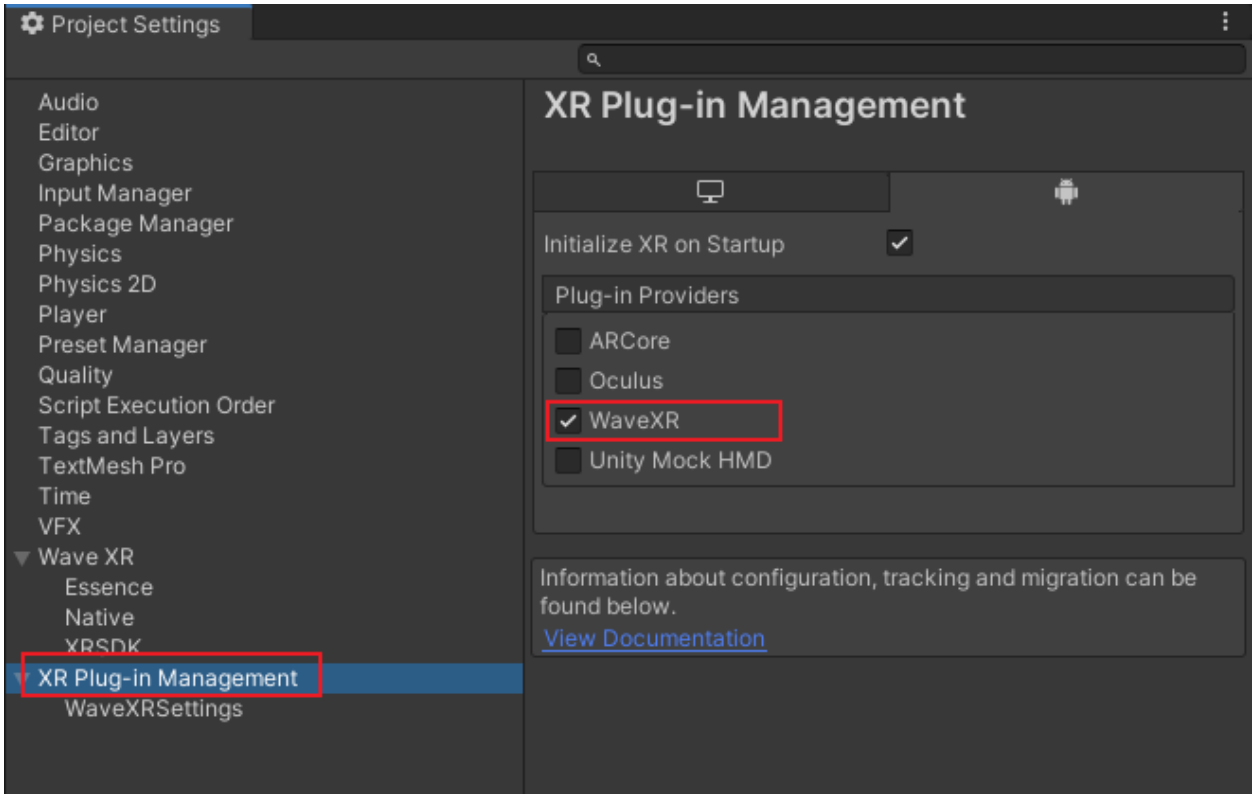


Select Scene file

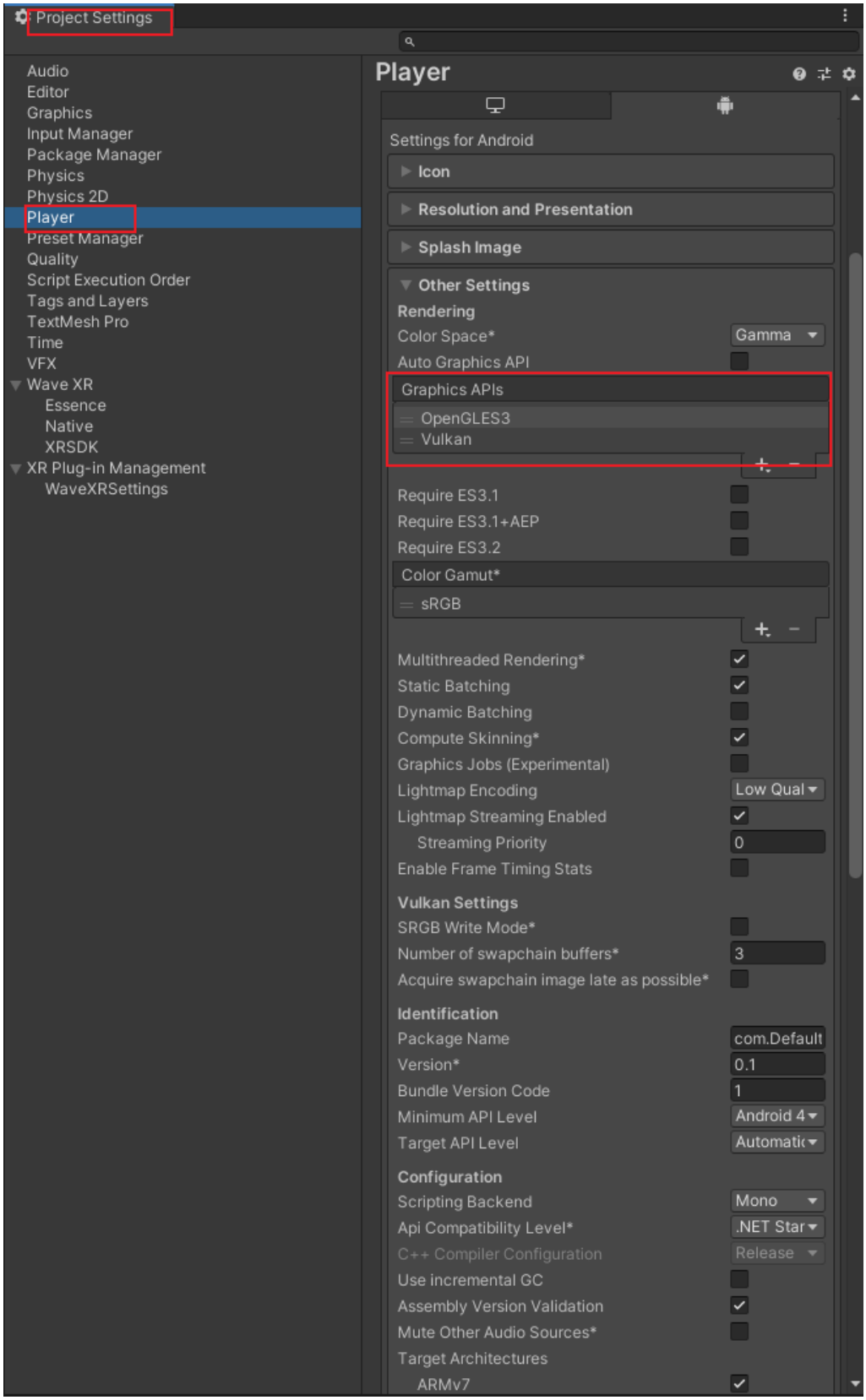


Build

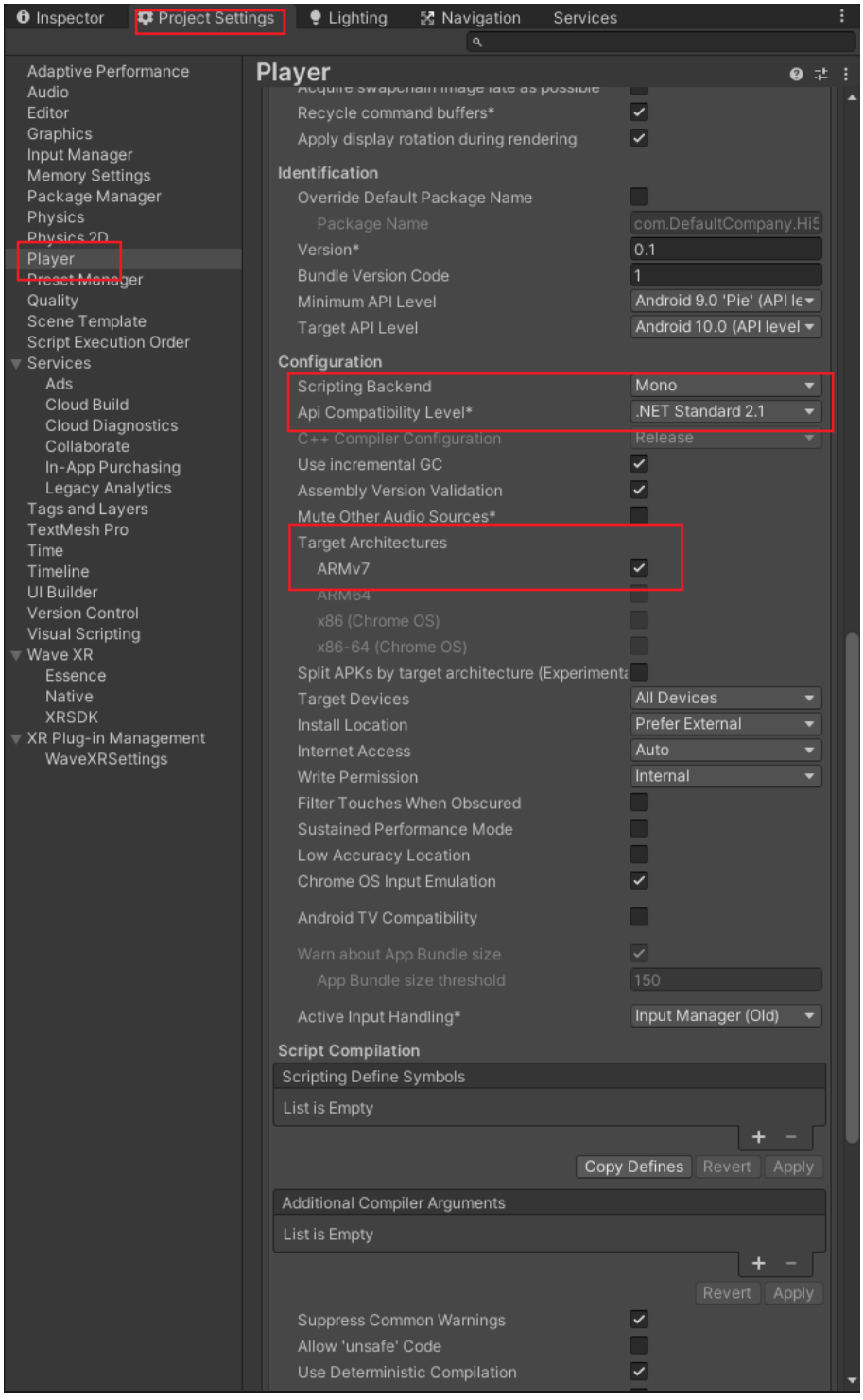
Switch XR platform



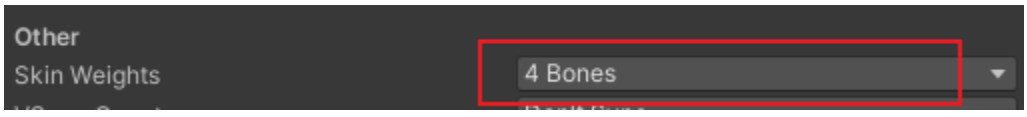
Focus does not support VULKAN



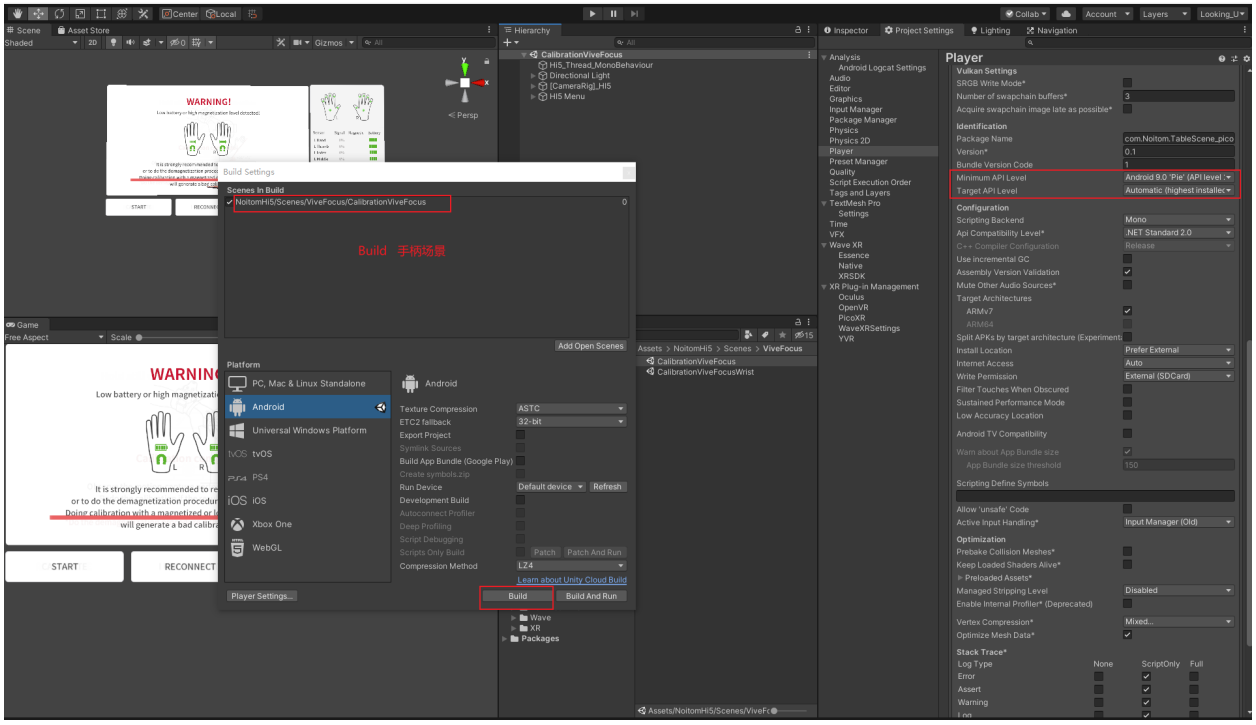
Set configuration



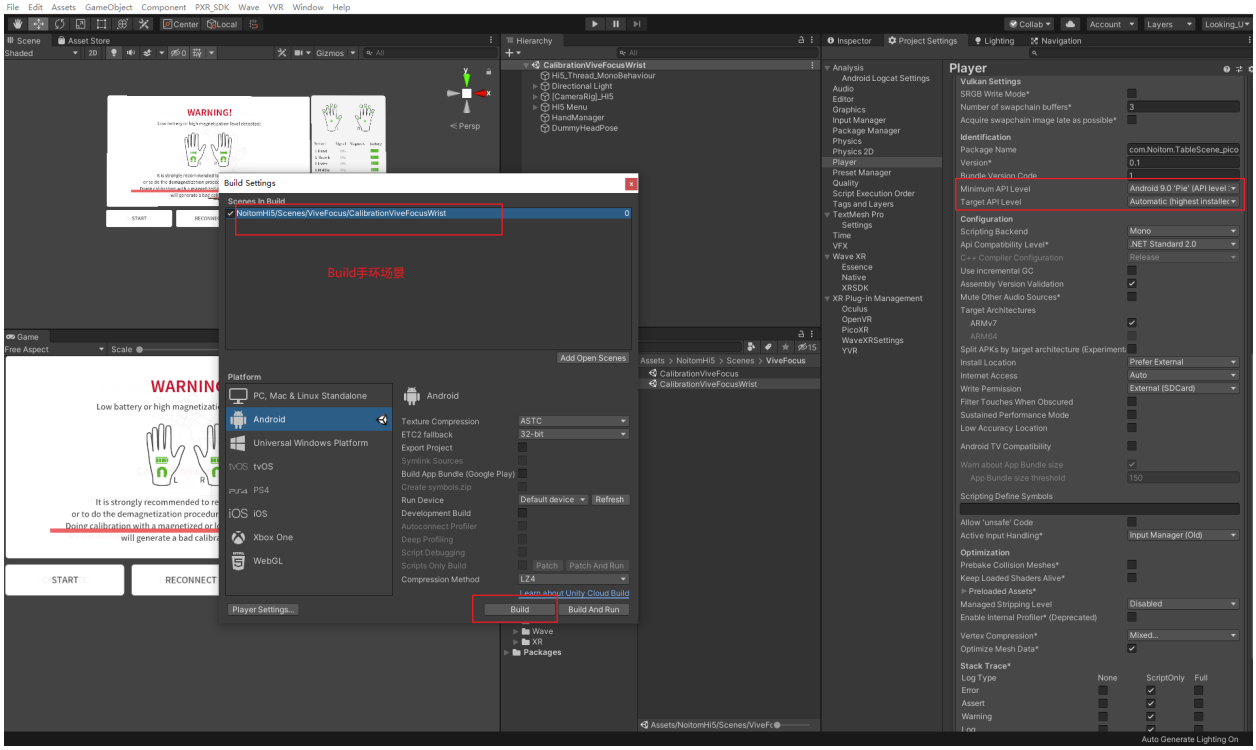
Set quality, Edit>Project Settings>Quality



Set the Android version, select the scene, and build



File Edit Assets GameObject Component PXR_SDK Wave VR Window Help



Interface usage

Get joint node data HI5_Glove_TransformData_Interface Script

Bone node data for each joint of the hand

```
// Obtain bone node data of the left hand joint
public Dictionary<EHi5_Glove_TransformData_Bones, Transform> GetLeftHandTransform()
{
return LeftHandBones;
}

// Get right hand joint bone node data
public Dictionary<EHi5_Glove_TransformData_Bones, Transform> GetRightHandTransform()
{
return RightHandBones;
}

// Hand joint point enumeration
public enum EHi5_Glove_TransformData_Bones
{
/// <summary>
/// The hand joint.
/// </summary> Hand = 0,
/// <summary>
/// The metacarpal joint of thumb finger.
/// </summary> HandThumb1,
/// <summary>
/// The proximal joint of thumb finger.
/// </summary> HandThumb2,
/// <summary>
/// The distal joint of thumb finger.
/// </summary> HandThumb3,
/// <summary>
/// The metacarpal joint of index finger.
/// </summary> InHandIndex,
/// <summary>
/// The proximal joint of index finger.
/// </summary> HandIndex1,
/// <summary>
/// The middle joint of index finger.
/// </summary> HandIndex2,
/// <summary>
/// The distal joint of index finger.
/// </summary> HandIndex3,
/// <summary>
/// The metacarpal joint of middle finger.
/// </summary>

InHandMiddle,
/// <summary>
```

```
/// The proximal joint of middle finger.
/// </summary> HandMiddle1,
/// <summary>
/// The middle joint of middle finger.
/// </summary> HandMiddle2,
/// <summary>
/// The distal joint of middle finger.
/// </summary> HandMiddle3,
/// <summary>
/// The metacarpal joint of ring finger.
/// </summary> InHandRing,
/// <summary>
/// The proximal joint of ring finger.
/// </summary> HandRing1,
/// <summary>
/// The middle joint of ring finger.
/// </summary> HandRing2,
/// <summary>
/// The distal joint of ring finger.
/// </summary> HandRing3,
/// <summary>
/// The metacarpal joint of pinky finger.
/// </summary> InHandPinky,
/// <summary>
/// The proximal joint of pinky finger.
/// </summary> HandPinky1,
/// <summary>
/// The middle joint of pinky finger.
/// </summary> HandPinky2,
/// <summary>
/// The distal joint of pinky finger.
/// </summary> HandPinky3,
/// <summary>
/// The number of joints of Hi5 bones.
/// </summary> NumOfHI5Bones,
}
```

Sensor Data

```
// Sensor enumeration
public enum EHi5_Glove_Sensor
{
Hand = 1, HandThumb, HandIndex, HandMiddle, HandRing, HandPinky
}
// Left Hand Sensor Information
private Dictionary<EHi5_Glove_Sensor, HI5SensorInfor> LeftHandBonesSensorInfor;
// Right hand sensor information
private Dictionary<EHi5_Glove_Sensor, HI5SensorInfor> RightHandBonesSensorInfor;

//HI5 Sensor Information
public class HI5SensorInfor
```

```
{
public HI5SensorInfor()
{
_magneticValue = 0;
_energyValue = 0;
_signalValue = 0;
}
// Get sensor magnetic state
public int MagneticValue { get { return _magneticValue; } set { _magneticValue = value; } }
// Get sensor power
public int EnergyValue { get { return _energyValue; } set { _energyValue = value; } }
// Get sensor signal
public int SignalValue { get { return _signalValue; } set { _signalValue = value; } }
internal int _magneticValue;
internal int _energyValue;
internal int _signalValue;
};
```

Hi5_Glove_Calibration_Process_Interface screenplay

Call the calibration command interface

```
/// <summary>
/// HI5 calibration pose.
/// </summary> public enum HI5_Pose
{
/// <summary>
/// Unknown pose.
/// </summary> Unknown = -1,
/// <summary>
/// Buddha Pose
/// </summary> BPose = 0,
/// <summary>
/// Pinch Pose.
/// </summary> PPose,

//APose,

//TPose, VPose = 4,
}

/*call sequence Vpos ->Bpos->Ppose*/
/// <summary>
/// Start calibration.
/// </summary>
/// <param name="pose">
/// The type of calibration pose by <see cref="HI5.HI5_Pose"/>.
/// </param>
public static void StartCalibration(HI5_Pose pose)
{
if (pose == HI5_Pose.BPose) isCalibratingBPose = true;

if (pose == HI5_Pose.PPose) isCalibratingPPose = true;
```

```
if (pose == HI5_Pose.VPose) HI5_Calibration.ResetCalibration();

CalibrationPose tranferPose = TransferPoseEnum(pose);


if (pose == HI5_Pose.BPose && HI5_Manager_Thread.Instance() != null)
HI5_Manager_Thread.Instance().AddCalibrationCommand(HI5_Operate_Command.HI5_Calibration_Send
_Data.ECalibr


if (pose == HI5_Pose.PPose && HI5_Manager_Thread.Instance() != null)
HI5_Manager_Thread.Instance().AddCalibrationCommand(HI5_Operate_Command.HI5_Calibration_Send
_Data.ECalibr

if (pose == HI5_Pose.VPose && HI5_Manager_Thread.Instance() != null)
HI5_Manager_Thread.Instance().AddCalibrationCommand(HI5_Operate_Command.HI5_Calibration_Send
_Data.ECalibr
// HI5_Device.StartCalibration(tranferPose);
}
```

Script

Get Calibration Progress

```
/// <summary>
/// Get the percent calibration.
/// </summary>
/// <param name="pose">
/// The type of calibration pose by <see cref="HI5.HI5_Pose"/>.
/// </param>
/// <returns>
/// The progress of the related calibration. The value is provided by percent number.
/// </returns>
public static int GetCalibrationProgress(HI5_Pose pose)
{
CalibrationPose tranferPose = TransferPoseEnum(pose);
int percent = 0;
if (HI5_Manager_Thread.Instance() != null)
{
percent = (int)HI5_Manager_Thread.Instance().Calibrationpercent;
}
if (pose == HI5_Pose.BPose && percent == 100)
{
//ruige 2018 11 5
//SaveBindTrackedObjectInfo();
isCalibratingBPose = false;
//HI5_Manager.GetGloveStatus().IsBposComplete = true;
//SetDefalutOffset();
}
if (pose == HI5_Pose.PPose && percent == 100)
```

```
{
//SaveCalibrationData();
//ruige 2018 11 5
//if (HI5_Log_Manager.Instance != null)
// HI5_Log_Manager.Instance.WriteLog();
//Debug.Log("Save Calibration Data " + value);
isCalibratingPPose = false;
}
if (pose == HI5_Pose.VPose && percent == 100)
{
//SaveCalibrationData();
//ruige 2018 11 5
//if (HI5_Log_Manager.Instance != null)
// HI5_Log_Manager.Instance.WriteLog();
//Debug.Log("Save Calibration Data " + value);
//isCalibratingPPose = false;
}
return percent;
//return HI5_Device.GetCalibratingPercent(tranferPose);
}
```