**NOLO VR Unity SDK**

**Documentation**

NOLO Co., Ltd

March 2018

目录

[1. Overview 3](#_Toc3356)

[1.1 About NOLO 3](#_Toc1558)

[2. Set Up Development Environment 3](#_Toc21981)

[3. Instructions 4](#_Toc7002)

[3.1 Quick Start 4](#_Toc15771)

[3.2 Debugging Instructions 7](#_Toc2847)

[4. API Description 8](#_Toc8300)

[4.1 Button Events 8](#_Toc22245)

[4.2 Touch Events 9](#_Toc16528)

[4.3 Vibration Events 10](#_Toc19391)

[4.4 Positional Information 10](#_Toc17231)

[4.5 Error Report 11](#_Toc31797)

[4.6 Connection Status of Device 11](#_Toc29169)

[5. Notes 12](#_Toc29157)

[5.1 Unity Setting 12](#_Toc18245)

[5.2 Set Origin 13](#_Toc7519)

[5.3 Set AppKey 13](#_Toc22162)

[5.4 Modify AndroidManifest.xml 14](#_Toc32224)

[5.5 Reset Orientation 14](#_Toc16110)

[5.6 Set Turn-around Key 15](#_Toc1999)

# Overview

## About NOLO

NOLO is dedicated to combine desktop-grade VR gaming experience with the convenience of mobile VR devices, redefining a mobile VR gaming experience like never before.

NOLO kit is compatible with some 87,000,000 VR headsets of all kinds currently on the market, indicating huge market potential. In addition, we’ve partnered with VR headset companies, robotic companies, and drone companies around the globe.

# Set Up Development Environment

Developers are required to prepare a Unity 5.6 or higher edition of NOLO VR Unity SDK. In case of debugging on the Android mobile phones, install NOLO HOME first, while apply for an Appkey on the NOLO Developer Platform and fill it in your Unity project.

Unity download address：

<https://unity3d.com/>

NOLO HOME download address：

<http://download.nolovr.com/download/nolohome.html>

You can use the public Appkey in the debugging phase, and change it to official Appkey when it is launched officially.

Public Appkey：4e4f4c4f484f4d457eff82725bc694a5

# Instructions

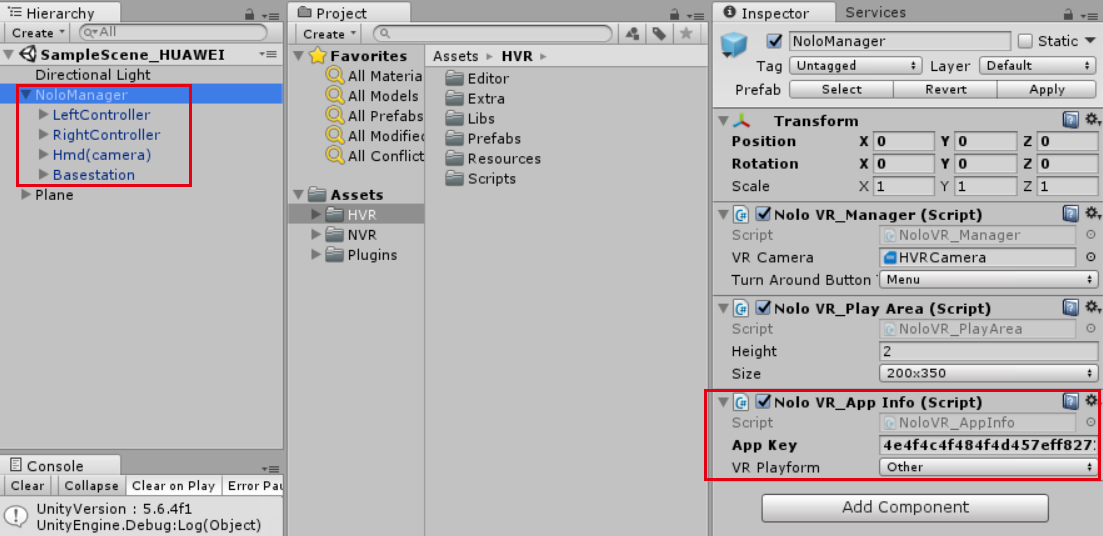
## Quick Start

#### （2）GearVR Project

1）Create a new Unity project and import the NOLO VR Unity SDK into it.

2）Create a new scenario and put NVR/Prefabs/NoloManager into it and save.

3）Fill the Appkey in the following location



4) Fill in the correct package name information to package the

settings and send it to the mobile phone or all-in-one to run.

## Debugging Instructions

Debug in Unity Editor: Only connect the NOLO headset marker to the computer with the USB cable, initiate the NOLO Assistant and turn on other NOLO devices. After the power information of all NOLO devices is displayed on the NOLO Assistant, click the Run button of Unity to debug in the Unity Editor.

Debug on Android clients: Install NOLO HOME on the mobile device, fill the correct Appkey in the project, and use the test key for debugging before the Appkey is reviewed. Also connect the NOLO headset marker with the OTG cable to the mobile phone or all-in-one. If it is prompted that "Run NOLO HOME to access USB device?", click OK to get the NOLO data in your APP.

Note: Currently only supports all-in-one using Qualcomm chips, such as iQiyi Adventure II, PICO G2.

# API Description

## Button Events

|  |  |
| --- | --- |
| **function name** | bool GetNoloButtonPressed() |
| **function description** | To check if a button is continuously being pressed down. (“pressed” status) |
| **input parameters** | Enum NoloButtonID |
| **return value** | bool |
| **prerequisites** | NoloVR\_Controller.GetDevice() |

|  |  |
| --- | --- |
| **function name** | bool GetNoloButtonDown() |
| **function description** | To check if a button is being pressed from “release” status. (“press” action) |
| **input parameters** | Enum NoloButtonID |
| **return value** | bool |
| **prerequisites** | NoloVR\_Controller.GetDevice() |

|  |  |
| --- | --- |
| **function name** | bool GetNoloButtonUp() |
| **function description** | To check if a button is being released from ‘pressed’ status. (“release” action) |
| **input parameters** | Enum NoloButtonID |
| **return value** | bool |
| **prerequisites** | NoloVR\_Controller.GetDevice() |

## Touch Events

|  |  |
| --- | --- |
| **function name** | bool GetNoloTouchPressed() |
| **function description** | To check if the touchpad is touched. (“touched” status) |
| **input parameters** | Enum NoloTouchID |
| **return value** | bool |
| **prerequisites** | NoloVR\_Controller.GetDevice() |

|  |  |
| --- | --- |
| **function name** | bool GetNoloTouchDown() |
| **function description** | To check if the touchpad is being touched. (“touch” action) |
| **input parameters** | Enum NoloTouchID |
| **return value** | bool |
| **prerequisites** | NoloVR\_Controller.GetDevice() |

|  |  |
| --- | --- |
| **function name** | bool GetNoloTouchUp() |
| **function description** | To check if the touchpad is being released. (“release” action) |
| **input parameters** | Enum NoloTouchID |
| **return value** | bool |
| **prerequisites** | NoloVR\_Controller.GetDevice() |

|  |  |
| --- | --- |
| **function name** | Vector2 GetAxis() |
| **function description** | To get the coordinates of the touched spot on the touchpad. |
| **input parameters** | Enum NoloTouchID: touchpad(default), other parameters are void (see appendix) |
| **return value** | Vector2 |
| **prerequisites** | NoloVR\_Controller.GetDevice() |

## Vibration Events

|  |  |
| --- | --- |
| **function name** | void TriggerHapticPulse() |
| **function description** | **To trigger controller vibration.** |
| **input parameters** | **Vibration intensity: 0~100 (int)** |
| **return value** | void |
| **prerequisites** | NoloVR\_Controller.GetDevice() |

## Positional Information

|  |  |
| --- | --- |
| **function name** | Nolo\_Transform GetPose() |
| **function description** | Get device position. |
| **input parameters** | Null |
| **return value** | Nolo\_Transform |
| **prerequisites** | NoloVR\_Controller.GetDevice() |

## Error Report

|  |  |
| --- | --- |
| **function name** | void ReportError () |
| **function description** | Log error messages. |
| **input parameters** | string |
| **return value** | void |
| **prerequisites** | NoloVR\_Playform.GetInstance() |

## Connection Status of Device

NoloVR\_Plugins.API.GetPoseByDeviceType(0).bDeviceIsConnected

The parameter 0 represents the headset marker, 1 represents the left controller, 2 represents the right controller, and 3 represents the base station. This method only applies to the Android platforms.

# Notes

## Set Origin

Turn on all NOLO devices, place the headset marker on the ground, press the button on the headset marker. The headset marker’s current position will be the origin in the game, aka the position of “NoloManager” in the game engine. The origin’s coordinates will be saved. This process only needs to be repeated if the Base Station has been moved.

## Set AppKey

A game must acquire an AppKey to run properly with NOLO CV1. An AppKey will be generated automatically when developers apply for their game on NOLO Developer Center. Please add NoloVR\_AppInfo script to your project workspace, and fill in the AppKey.

When the game does not upload NOLO HOME, you can use this public Appkey for development testing.

Public Appkey：4e4f4c4f484f4d457eff82725bc694a5

## Modify AndroidManifest.xml

Add the following scripts in AndroidManifest.xml:

<uses-permission android:name="android.permission.BROADCAST\_STICKY" />

<uses-permission android:name="nolo.permission.ACCESS\_SERVER" />

<uses-permission android:name="android.permission.PACKAGE\_USAGE\_STATS" />

<uses-permission android:name="android.permission.SYSTEM\_ALERT\_WINDOW" />

<uses-permission android:name="android.permission.ACCESS\_WIFI\_STATE" />

<uses-permission android:name="android.permission.INTERNET" />

<uses-permission android:name="android.permission.ACCESS\_NETWORK\_STATE" />

<uses-permission android:name="com.android.launcher.permission.WRITE\_SETTINGS" />

<uses-permission android:name="android.permission.WRITE\_APN\_SETTINGS" />

Add the following scripts in Application：

<meta-data android:name="com.picovr.type" android:value="vr"/>

Add the following scripts in Activity：

<intent-filter>

<action android:name="android.intent.action.NOLOHOME" />

<category android:name="android.intent.category.DEFAULT" />

</intent-filter>

## Reset Orientation

Upon starting a game, if the forward direction in the game does not point towards Nolo Base Station, or the controller orientation seems a little odd, you may need to reset orientation by doing the following: Put on your headset, face the Nolo Base Station, point both controllers towards the Nolo Base Station, then double click the power button on either controller.

## Set Turn-around Key

You may assign a button on the controller to be a “turn-around” hotkey as shown in the figure. When you’re in a game, you can double click that button to turn your view angle by 180 degrees in an instant( figure 4).



figure 4