**NOLO VR Unity SDK**

**Documentation**

NOLO Co., Ltd

March 2018

CONTENTS

[1. Overview 3](#_Toc510201222)

[1.1 About NOLO 3](#_Toc510201223)

[1.2 About NOLO CV1 3](#_Toc510201224)

[1.3 About NOLO HOME 4](#_Toc510201225)

[1.4 About NOLO VR UE4 SDK 4](#_Toc510201226)

[1.5 NOLO VR UE4 SDK Framework 4](#_Toc510201227)

[2. Set Up Development Environment 6](#_Toc510201228)

[3. API Description 7](#_Toc510201229)

[3.1 Button Events 7](#_Toc510201230)

[3.2 Touch Events 8](#_Toc510201231)

[3.3 Vibration Events 9](#_Toc510201232)

[3.4 Positional Information 9](#_Toc510201233)

[3.5 Error Report 10](#_Toc510201234)

[4. Notes 10](#_Toc510201235)

[4.1 Set Origin 10](#_Toc510201236)

[4.2 Set AppKey 10](#_Toc510201237)

[4.3 Modify AndroidManifest.xml 11](#_Toc510201238)

[4.4 Set Camera Parameters 11](#_Toc510201239)

[4.5 Reset Orientation 12](#_Toc510201240)

[4.6 Set Turn-around Key 12](#_Toc510201241)

# 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.

## About NOLO CV1

NOLO CV1 is a short-range high-accuracy motion tracking equipment kit for VR/AR gaming systems, composed of a Base Station, a headset marker and 2 controllers.

NOLO CV1 is compatible with all mainstream mobile VR headsets, some PC VR/AR headsets, and All-In-One’s based on NibiruOS, providing position information and interaction functionality to these devices. With simple setup at first launch, users can move around in the virtual world like in real world, and interact with virtual objects using our hand controllers.

## About NOLO HOME

NOLO HOME is the first ever 6-DoF mobile VR gaming platform, developed by NOLO. With NOLO HOME, NOLO CV1, and a decent smartphone, users can enjoy any VR content (e.g. a VR game on your phone) whenever, wherever, wirelessly, creating a whole new way of entertainment accessible to and affordable for all.

NOLO HOME Android version is now available to the public, iOS version is currently under development.

## About NOLO VR UE4 SDK

NOLO VR UE4 SDK is developed by NOLO Co., Ltd, to provide data like positional information from NOLO devices for UE4 developers. With this SDK, you can develop 6-DoF mobile VR games that run on smartphones or GearVR(GearVR SDK required).

## NOLO VR UE4 SDK Framework

NOLO VR UE4 SDK incorporates a C/S framework. NOLO HOME (server) reads and processes data obtained directly from NOLO CV1 devices. Game (client) establishes a connection with NOLO HOME via AIDL to obtain data from NOLO CV1 (through SDK/NOLO HOME). Therefore, the game app needs NOT read any data directly from USB cable (figure 1).

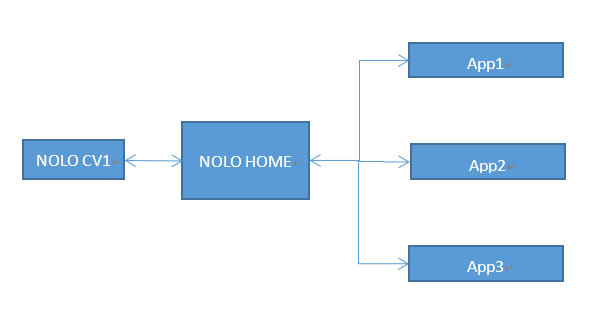


figure 1

NOLO VR Unity SDK and NOLO VR UE4 SDK are built upon NOLO VR Android SDK. NOLO VR Android SDK establishes a communication cannel with NOLO HOME via AIDL, to obtain data from NONO CV1 (figure 2).

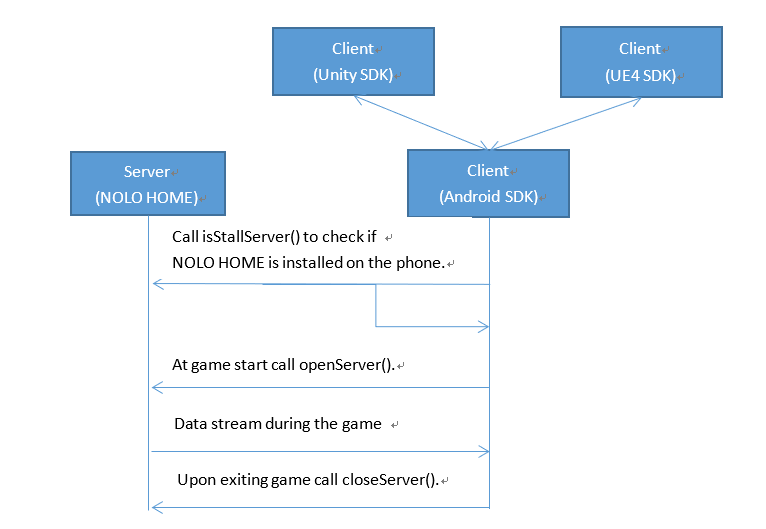


figure 2

# Set Up Development Environment

Requires Unity 5.6 or above and NOLO VR Unity SDK.

Requires NOLO Driver For Windows (if you work on Windows).

Requires NOLO HOME app on your test phone(Android).

# 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() |

# 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.

## 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" />

## Set Camera Parameters

(figure 3)

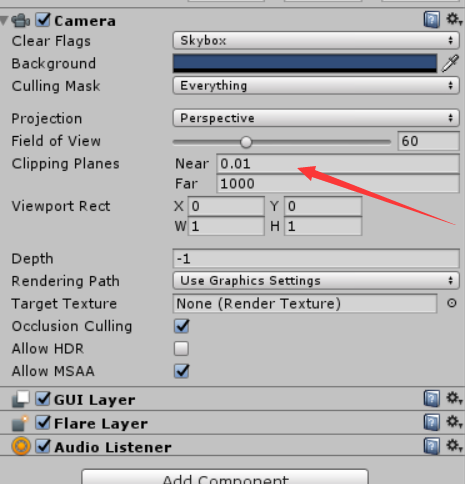


figure 3

## 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