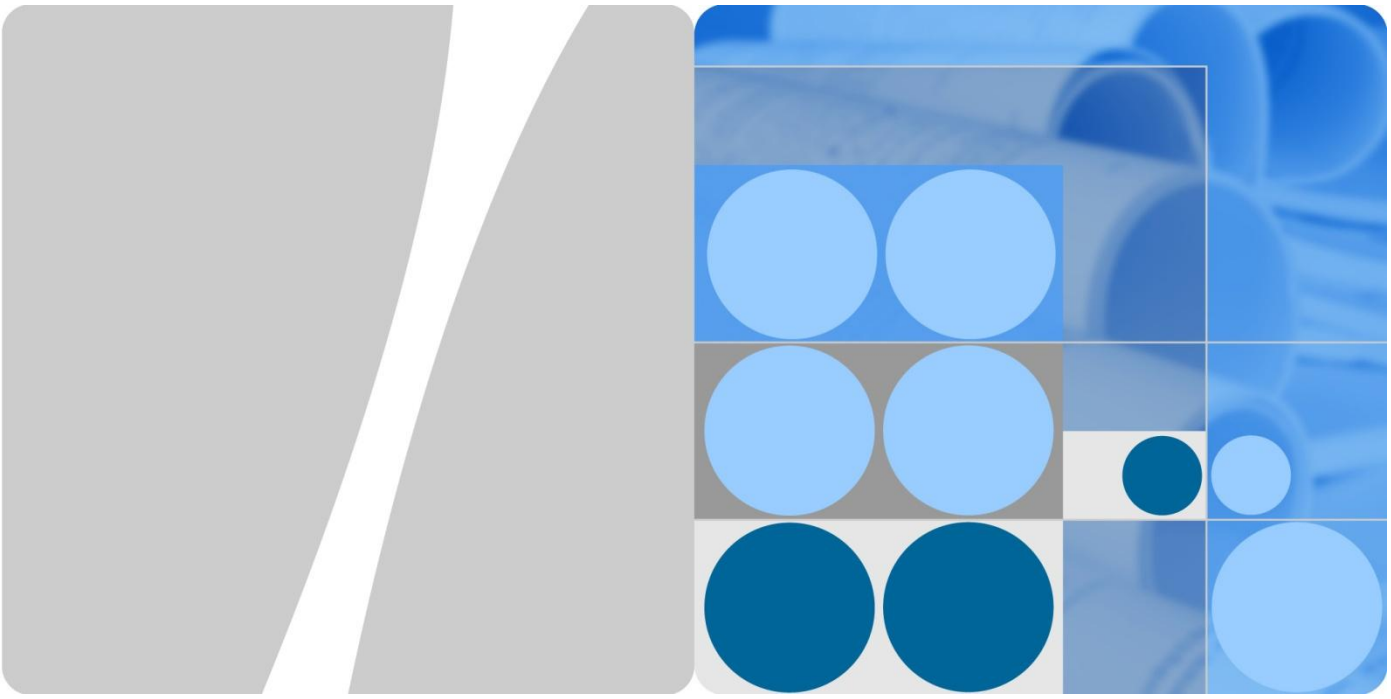


HuaweiVR SDK2.0 Development Guide

Issue 2.0.0

Date 2017-09-08

HUAWEI TECHNOLOGIES CO., LTD.

Trademarks and Permissions


and other Huawei trademarks are trademarks of Huawei Technologies Co., Ltd.

All other trademarks and trade names mentioned in this document are the property of their respective holders.

Notice

The purchased products, services and features are stipulated by the contract made between Huawei and the customer. All or part of the products, services and features described in this document may not be within the purchase scope or the usage scope. Unless otherwise specified in the contract, all statements, information, and recommendations in this document are provided "AS IS" without warranties, guarantees or representations of any kind, either express or implied.

The information in this document is subject to change without notice. Every effort has been made in the preparation of this document to ensure accuracy of the contents, but all statements, information, and recommendations in this document do not constitute a warranty of any kind, express or implied.

Huawei Technologies Co., Ltd.

Address: Huawei Industrial Base
Bantian, Longgang
Shenzhen 518129
People's Republic of China

Website: http://www.huawei.com

Email: support@huawei.com

Change History

Date	Version	Description
2017-09-08	2.0.0	This is the second edition.

Contents

1 Overview

1.1 Function Introduction

HuaweiVR SDK2.0 and Unity are used to develop HuaweiVR applications. HuaweiVR SDK2.0 for Unity (SDK2.0 for short) provides the following functions:

- Customizes application scenes.
- Obtains the helmet posture and sensor data.
- Obtains the controller posture, power, and sensor data.
- Locks and unlocks the helmet posture.
- Automatically displays a dialog box of notification messages, such as calls, short messages, and clocks.
- Provides controller shortcut buttons.
- Calibrates dispersion.

1.2 Basic Knowledge Preparation

Before using the SDK2.0, you need to have a basic understanding of the Unity platform and C# language.

1.3 Supported Platforms

- Smartphone: Huawei flagship smartphones, such as Mate 10, Mate 9 Pro, and P9 Plus.
- OS: EMUI5.0 or later (Android N or later)
- Unity version: 5.5 or later

2 Development Guide

The Unity platform in 5.5 or later is used to construct Android-based HuaweiVR applications. This guide describes how to set Unity and develop and construct HuaweiVR applications.

2.1 Preparation

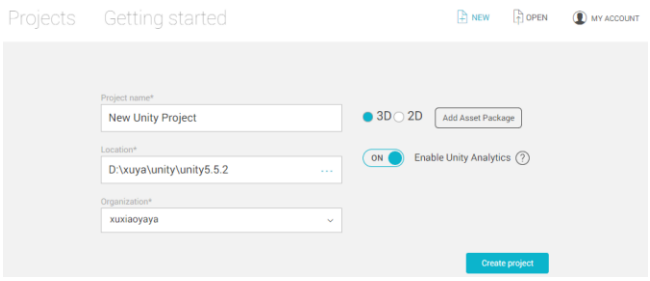
- You have installed the Unity platform on your PC.
- You have downloaded the compressed package **HuaweiVR SDK2.0 For Unity**.
- You have installed the HuaweiVR SDK service APK and the controller service APK on the smartphone.
- You have prepared the Huawei smartphone and HuaweiVR helmet.

2.2 Process

2.2.1 Creating a Project and Importing HvrSdk.unitypackage on the Unity

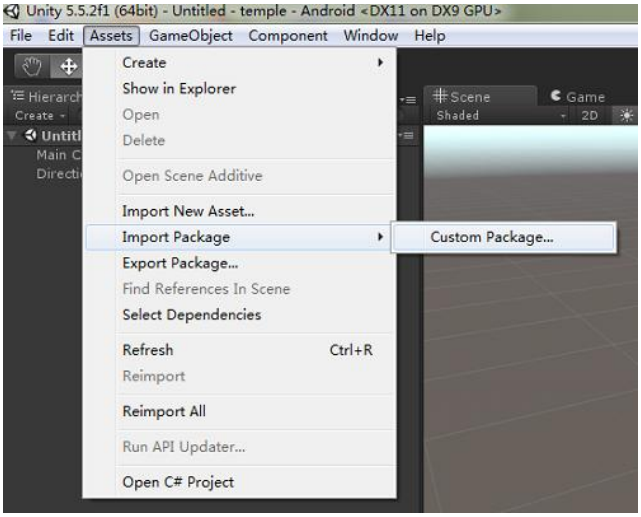
Step 1 Log in to the Unity platform and create a 3D project.

Figure 2-1 Getting started page on the Unity platform



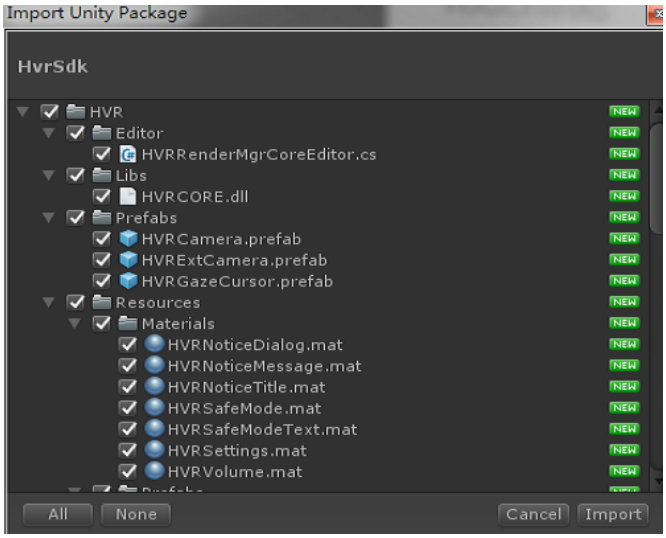
Step 2 Choose **Assets** > **Import Package** > **Custom Package**.

Figure 2-2 Page for importing HvrSdk.unitypackage



Step 3 In the displayed **Import Unity Package** dialog box, select **HvrSdk.unitypackage** that you have downloaded and click **Import**.

Figure 2-3 Import Unity Package dialog box



- After **HvrSdk.unitypackage** is imported, **HVRAddUILayer.cs** in **Assets/HVR/Scripts** automatically adds HVRUILayer and HVRVolumeUILayer for drawing the gaze cursor and volume setting box, respectively. If UI layers fail to be automatically added, a dialog box is displayed, asking you to manually add HVRUILayer and HVRVolumeUILayer.

Figure 2-4 Layers page

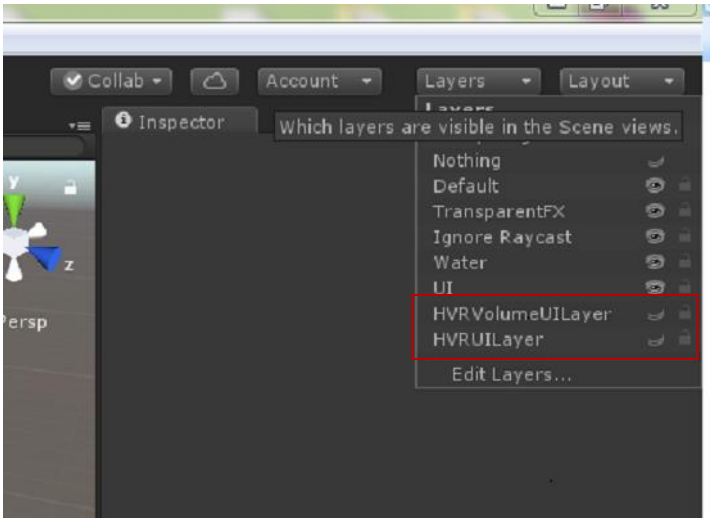
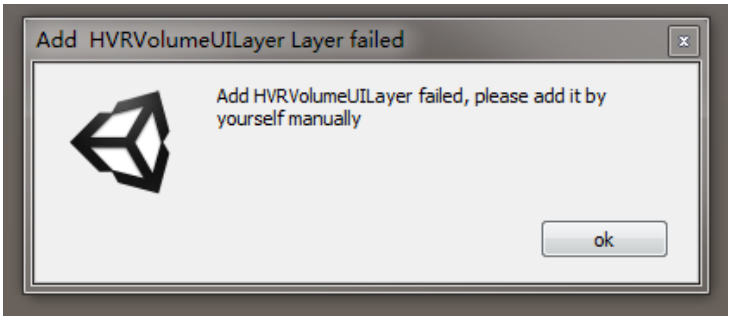


Figure 2-5 Dialog box displayed after adding UI layers fails



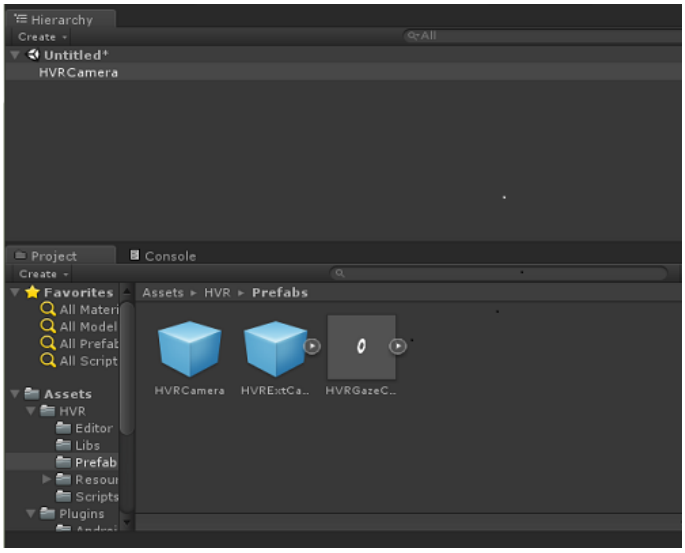
- You are not allowed to draw other contents on the two UI layers. In addition, the two UI layers are not displayed for other cameras in the Unity scene. For example, after **Culling Mask** is modified for the left and right cameras of SDK2.0, the contents of the two UI layers are displayed.

----End

2.2.2 Adding a HuaweiVR Camera to an Application Scene

To enable the VR mode to be displayed in an application scene, you need to add a HuaweiVR camera named **HVRCamera**, which is saved in the **Assets/HVR/Prefabs** directory. Click and hold **HVRCamera** and drag it to the **Hierarchy** area. **MainCamera** automatically generated during project creation is deleted.

Figure 2-6 Adding HVRCamera



After **HVRCamera** is added, the VR mode is displayed in the application scene on the smartphone. If you click **HVRCamera**, the camera attribute is displayed in the **Inspector** area, where:

- **Render Depth Format** indicates the depth parameter when **RenderTexture** is initialized on the Unity. The default value is 24 bit. It can be set to 16 bit, 24 bit, or 32 bit.
- **Render Resolution Scale** indicates the resolution scale when **RenderTexture** is initialized on the Unity. The default value is 1, and the default resolution is 1080 x 1080. You can set the rendering resolution based on scene complexity.

NOTE

Modifying the resolution changes the content definition and affects the performance and power consumption.

Figure 2-7 Render Depth Format and Render Resolution Scale settings

2.2.3 Integrating Controllers

Perform the following operations to interact SDK2.0 with controllers:

- Obtain controller handles.

  IControllerHandle controllerHandle = HvrApi.GetControllerHandle ();

HuaweiVR SDK is initialized by the Awake() function. You are advised to call SDK functions using the Start() function.

- Obtain serial numbers of available controllers.

  int[] indices = controllerHandle.GetValidIndices ();

  indices[0] indicates a standard input event used for the touchpad of the HuaweiVR1.0 helmet. **indices[1]** indicates a controller event. **indices[2]** and later indicate other new controllers.

- Obtain controller objects based on the serial numbers.

  Obtain the helmet information and determine whether the helmet type is HuaweiVR1.0 or HuaweiVR2.0. For details about the helmet type, see section 4.5.10 "GetHelmetInfo" in *HuaweiVR SDK2.0 Unity API Description*.

  You can determine the interaction mode based on the helmet type.

  - When the helmet type is HuaweiVR1.0
    - If no controller is connected, use the touchpad and obtain the controller object of **indices[0]**:

      IController controller = controllerHandle.GetControllerByIndex(indices[0]);

    - If a controller is connected, use the controller and obtain the controller object of **indices[1]**:

      IController controller = controllerHandle.GetControllerByIndex(indices[1]);

  - When the helmet type is HuaweiVR2.0
    - Use the controller and obtain the controller object of **indices[1]**. The touchpad is not supported.

      IController controller = controllerHandle.GetControllerByIndex(indices[1]);

    - Wait for controller connections if no controller is connected.

- Determine the controller connection status.

  After obtaining controller objects, determine whether a controller is connected. If it is, perform subsequent operations.

  bool status = controller.IsAvailable ();

Description of controller functions:

- Controllers are compatible with all functions (such as left-swipe, right-swipe, swipe-up, swipe-down, click, touch-down, touch-up, and back) of a touchpad from the helmet in HuaweiVR1.0.
- HuaweiVR2.0 supports the following controller functions:
  - Obtaining the controller posture
  - Notifying the controller connection and disconnection
  - Obtaining click events of the touchpad, back button, and trigger button (excluding the home and volume buttons)
  - Obtaining position and swipe events of the touchpad
- To obtain the long press and short press events of the controller buttons, refer to the implementation of the example code Assets\HVR\Scripts\HVRControllerUtils.cs.
- Description of default shortcut buttons:

Operation	Button	Button Holding Duration	Function
Click	Home	< 1s	Navigate to the **VRLauncher** page.
Long press	Home	≥ 1s	Reset the viewpoint.
Long press	Back	≥ 3s	Navigate to the **VRSetting** page.
Long press	Home and Trigger	≥ 1s	Capture a screenshot.

- If you need to customize shortcut buttons, ensure that the customized buttons do not conflict with the preceding shortcut buttons.

NOTE

The controller status is updated in Update(). If you want to ensure that all the scripts that get the controller status in the application have the same controller status acquired in each frame, developers need to manually adjust the script execution order to ensure that these scripts all are executed before or after the SDK script:

Tip: Manually set the script execution order: Edit -> Project Settings -> Script Execution Order menu items, set the script to achieve digital.

2.2.4 Configuring AndroidManifest.xml of Applications

The **AndroidManifest.xml** file of applications must meet the following requirements:

- The following meta-data is included in an application to indicate that the application is a HuaweiVR application.

```
<meta-data android:name="com.huawei.android.vr.application.mode"
android:value="vr_only"/>
```

- The following meta-data is included in an application to ensure proper display for smartphones with special screen proportions, such as Mate 10 Pro.

```
<meta-data android:name="android.max_aspect" android:value="2.1" />
```

- intent-filter in the VR activity is modified as follows to remove desktop icons and provide switching portals in the VRLauncher.

```
<activity android:name="yourPackage.yourActivity">
    <intent-filter>
        <action android:name="com.huawei.android.vr.action.MAIN"/>
        <category android:name="android.intent.category.DEFAULT"/>
    </intent-filter>
</activity>
```

- The **AndroidManifest.xml** file contains the following permission for using the system VR service.

```
<uses-permission android:name="com.huawei.android.permission.VR"/>
```

- The **AndroidManifest.xml** file contains the following permission for connecting to the controller service.

```
<uses-permission
android:name="com.huawei.vrhandle.permission.DEVICE_MANAGER" />
```

- The **AndroidManifest.xml** file contains the following permission for capturing screenshots.

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

For details, see the default **AndroidManifest.xml** file in the **Assets\Plugins\Android** directory.

- In addition, if the developer needs to debug directly on the phone, you need to include the following meta-data in an application, but the official release of the application **must remove the tag**, this feature is only used for debugging .

```
<meta-data android:name="com.huawei.android.vr.sensor.mode"
android:value="mobile"/>
```

2.2.5 VRLauncher Icon Requirements

If you need to display application icons in VRLauncher, set icons according to the following specifications:

- Name: vr_icon.png
- Size: less than 200 KB
- Format: .png
- Resolution: 576 x 324
- Icon storage path: **Assets/StreamingAssets/vr_icon.png** in the 3D project on the Unity platform

2.2.6 Compiling Configuration Items

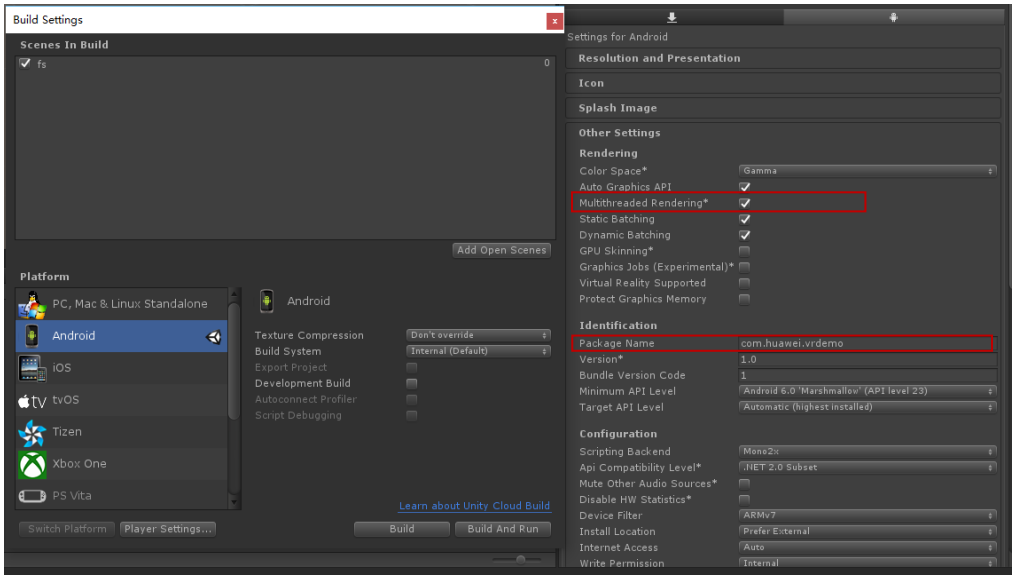
Step 1 On the Unity, choose **File** > **Build Settings…**. In the displayed **Build Settings** dialog box, select **Android** in the **Platform** area.

Step 2 Click **Player Settings…**. On the displayed **Inspector** page, select **Multithreaded Rendering** in the **Other Settings** area.

Step 3 SDK2.0 does not support **Vulkan** temporarily. It is recommended to check the **Auto Graphics API** option.

Step 4 Set a value for **Package Name**, such as **com.huawei.vrdemo**.

Figure 2-8 Building Settings page



Step 5 In the **Build Settings** dialog box, click **Build** to generate an APK. Alternatively, connect a PC to the smartphone and click **Build And Run** to download and open the APK on the smartphone.

----End

2.2.7 Customizing an Activity

If customized activities are available in compiled Android-based applications, **HVRActivity** in HuaweiVR Unity SDK needs to be inherited.

HVRActivity is defined in the **hvrbridge.jar** file in the **Assets\Plugins\Android** directory and is inherited from **UnityPlayerActivity**.

Perform the following operations in an Android project:

```
import com.huawei.vrlab.HVRActivity;
public class MainActivity extends HVRActivity {
.....
}
```

In addition, change the activity name to a customized activity name in the **AndroidManifest.xml** file, and ensure that intent-filter meets the requirements in section 2.2.4 "Configuring AndroidManifest.xml of Applications."

2.2.8 Setting Prompt Information on the 2D Page

SDK2.0 supports the prompt function on the 2D page. The 2D page is contained in **Assets\Plugins\Android\hvrprompt.aar** and is packaged to the application by default.

This function is required only when a 2D activity is available in an application and a 2D activity needs to be dynamically switched to the VR activity.

You can use the following example codes in your 2D activity to enable the prompt information display. After connecting the helmet, you can start your own VR activity. VRLauncher is skipped.

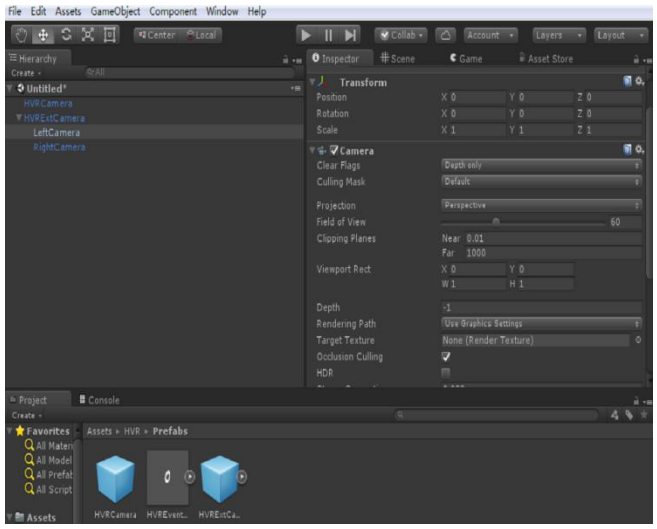
```
Intent intent = new Intent();
intent.setPackage(getPackageName());
intent.setAction("com.huawei.android.vr.PROMPT");
startActivity(intent);
```

2.2.9 Integrating Multiple Cameras

If multiple cameras on the Unity need to be integrated, perform the following operations to add the cameras:

Drag **HVRExtCamera** from the **Assets/HVR/Prefabs** directory to the **Hierarchy** area. Under **HVRExtCamera**, two cameras are mounted: **LeftCamera** and **RightCamera**, as shown in Figure 2-9. **LeftCamera** and **RightCamera** correspond to the left and right views of the HuaweiVR helmet. You can drag multiple groups of cameras named **HVRExtCamera** and modify the parameters of each group of the cameras, such as **Clear Flags**, **Culling Mask**, and **Depth**.

Figure 2-9 Multi-camera prefab setting page



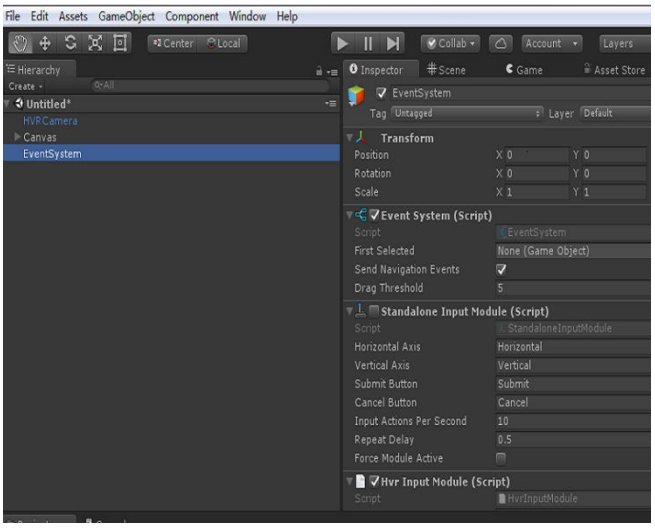
2.2.10 Integrating Gaze Cursor Prefabs (Only for HuaweiVR1.0)

The gaze cursor prefab applies only to HuaweiVR1.0. Interaction based on gaze cursors is not recommended for HuaweiVR2.0. Instead, interaction based on controllers is recommended.

Perform the following operations to integrate gaze cursor prefabs:

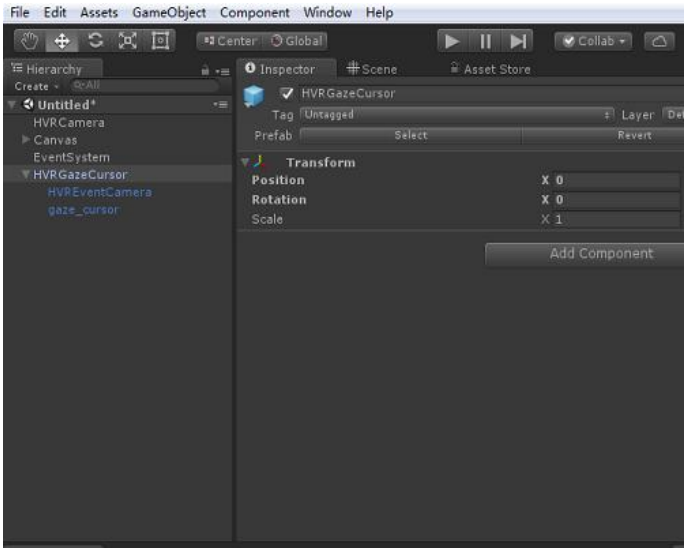
Step 1 Bind **HvrInputModule.cs** with **EventSystem** and deselect **Standalone Input Module.cs**.

Figure 2-10 Binding HvrInputModule.cs



Step 2 Drag **HVRGazeCursor** from the **Assets/HVR/Prefabs** directory to the **Hierarchy** area. Under **HVRGazeCursor**, images **HVREventCamera** and **gaze_cursor** are mounted. You can replace the images.

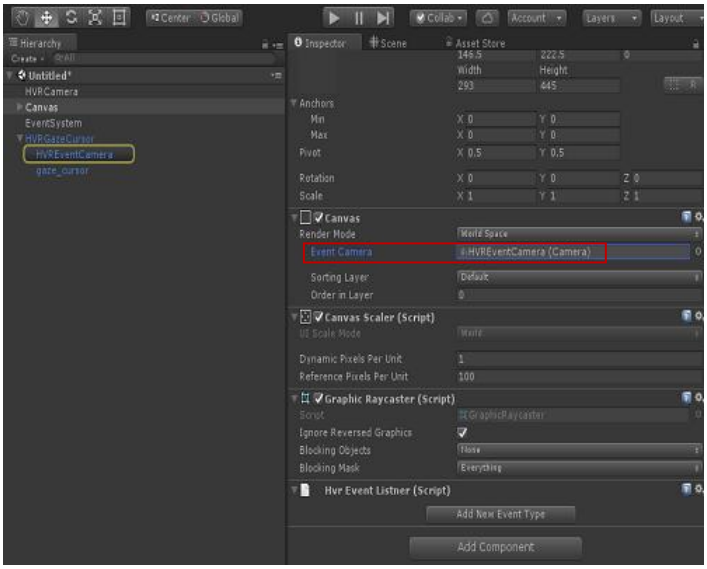
Figure 2-11 HVREventCamera setting page



Step 3 Use **HVREventCamera** for listening to UI events. Before you use it:

- Set **Render Mode** to **WorldSpace** in **Canvas** in the **Hierarchy** area.
- Set **Event Camera** to **HVREventCamera** in **Canvas** in the **Inspector** area.
- Bind **HvrEvenListner.cs** with **Canvas**. This script provides static delegation variables, including **onEnter**, **onExit**, **onDown**, **onUp**, and **onClick**. These variables are used to transfer to-be-listened UI input events of a gaze cursor.

Figure 2-12 Canvas setting page



----End

3 FAQs

3.1 Can SDK2.0 Be Used on Other Non-Huawei SmartPhones?

No, it cannot. SDK2.0 is developed for Huawei smartphones. You need to select Huawei smartphones for development because system layers of Huawei smartphones are optimized a lot.

3.2 Can SDK2.0 Be Used on Any Huawei Smartphone?

No, it cannot. SDK2.0 is developed for HuaweiVR helmets and related-model Huawei smartphones.

3.3 Which Version of the Unity Platform Is Used for Development?

You are advised to use the Unity platform in 5.5.0 or later.

3.4 Can HuaweiVR Applications Be Executed If the Smartphone Is Not Connected to the Helmet?

Yes, please configure in the following two ways:

1. run the **adb shell setprop hvr.imm.mobile enable** command.
2. Refer to 2.2.4, under the application of AndroidManifest contains meta-data:**<meta-data android:name="com.huawei.android.vr.sensor.mode" android:value="mobile"/ >** , but the tag must be removed from the officially released application. This feature is only for debugging.

3.5 Can Applications Be Started Using Other Methods Besides Being Started Through the VRLauncher Page?

Yes. You can run **adb shell am start -n** *Application package name*/**com.huawei.vrlab.HVRModeActivity** to navigate to the 2D prompt page and connect the smartphone to the helmet. Then, the application is started. This method applies only to HuaweiVR2.0.

3.6 Does the HuaweiVR Service APK Need to Be Updated After HvrSdk.unitypackage Is Updated?

Yes. After **HvrSdk.unitypackage** is updated, you need to install the matching HuaweiVR SDK service APK **HVRSDKServerAppSigned.apk** on the smartphone.

3.7 Why Is a Controller Not Connected?

1. Check whether the controller battery is installed properly.
2. Check whether the controller service APK is installed, running properly, and matches the controller.
3. Check whether the controller is working properly using the Demo.

3.8 developers only need to integrate HvrSdk.unitypackage to publish applications to mobile phones to run?

No, you need to make sure that HVRSDKServerAppSigned.apk is installed on your phone.

3.9 How do developers bind objects under camera components?

Please refer to Section 4.1.2 sample code in the **HuaweiVR SDK2.0 Unity API** Documentation.