PicoVRAndroidSDK_Unity

Development Document

Version：v_2.2.9

Beijing Pico Technology Co., Ltd.

Directory

1 Preface

1.1 Purpose

The purpose for creating this development manual is to help the developers to develop Unity Apps using the "PicoVR Android SDK for Unity".

Target users: technicians, Unity developers.

1.2 Background

PicoVRAndriodSDK_Unity is a Software Development Kit, which is released by Beijing Pico Technology Co., Ltd. It is intended for Unity 3D software developers to develop applications suitable for Pico VR HMDs. The SDK mainly supports the functions of sensor fusion, binocular stereo rendering, optical distortion correction, asynchronous time warping, single buffer rendering, multi-interactive support 3DoF/6DoF, 3D stereo sound, power and radiating management, account and payment management, etc.

The SDK in this documentation will be released as Unity packages. Before accessing the SDK, the Unity development environment needs to be configured, and then the development package can be imported. We provide some demos in the Assets\Pvr_UnitySDK\Scenes\Examples folder to developers for reference.

2 Supported Devices

2.1 VR Devices

	Product
Pico	Pico 1 & Pico 1s & Pico U
	Pico Neo DK/ DKS & Goblin

3 Developing Environment

Software	Version Requirement
JDK	jdk1.7.0_01 and later
Android SDK	API Level 19 and later

3.1 Supported Unity Versions

Unity5.2 and later version(nonsupport Unity2017).

Attentions:

- In some cases, the compiled application may be abnormal, unless the option "Use 32-bit Display buffer" is selected.
- The FPS may be different in the applications compiled by different Unity versions.
- UGUI does not work well in Unity5.3.5f1, but Unity5.3.5 P5 fix it and works well.
- In Unity Player Setting **do not** select Multithreaded Rendering option.

- In Unity5.4 and later versions, you need to modify the vertical sync settings.

  打开 Edit 菜单

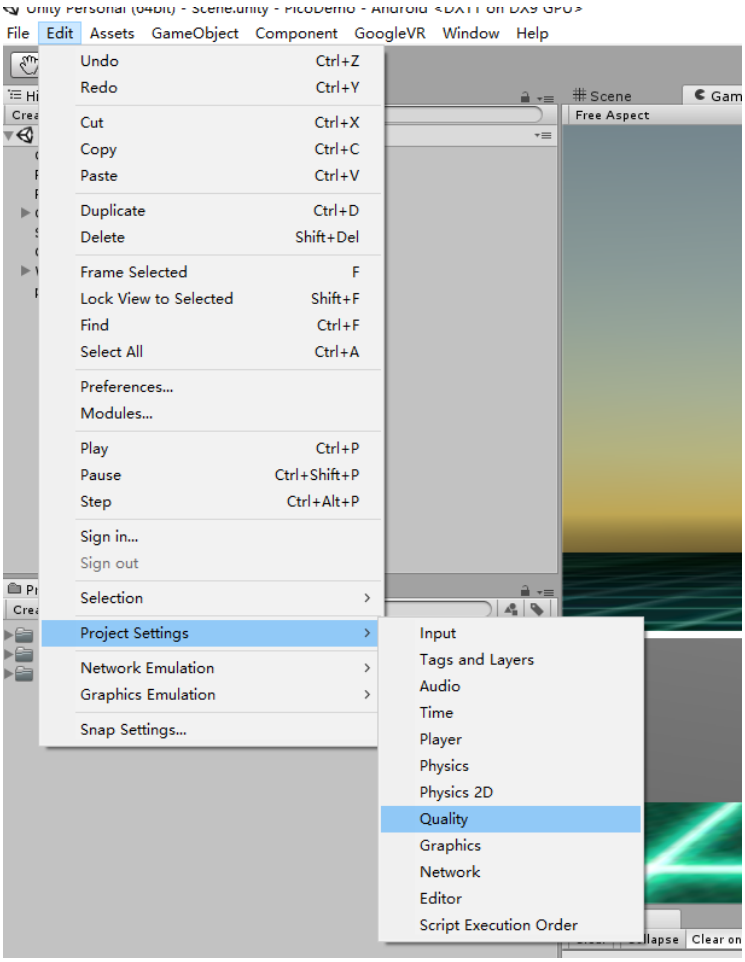


图 3.1.1

Edit->Project Setting->Quality

Select the Simple Levels, in the **Other**, and change the V sync Count, select Don`t Sync.

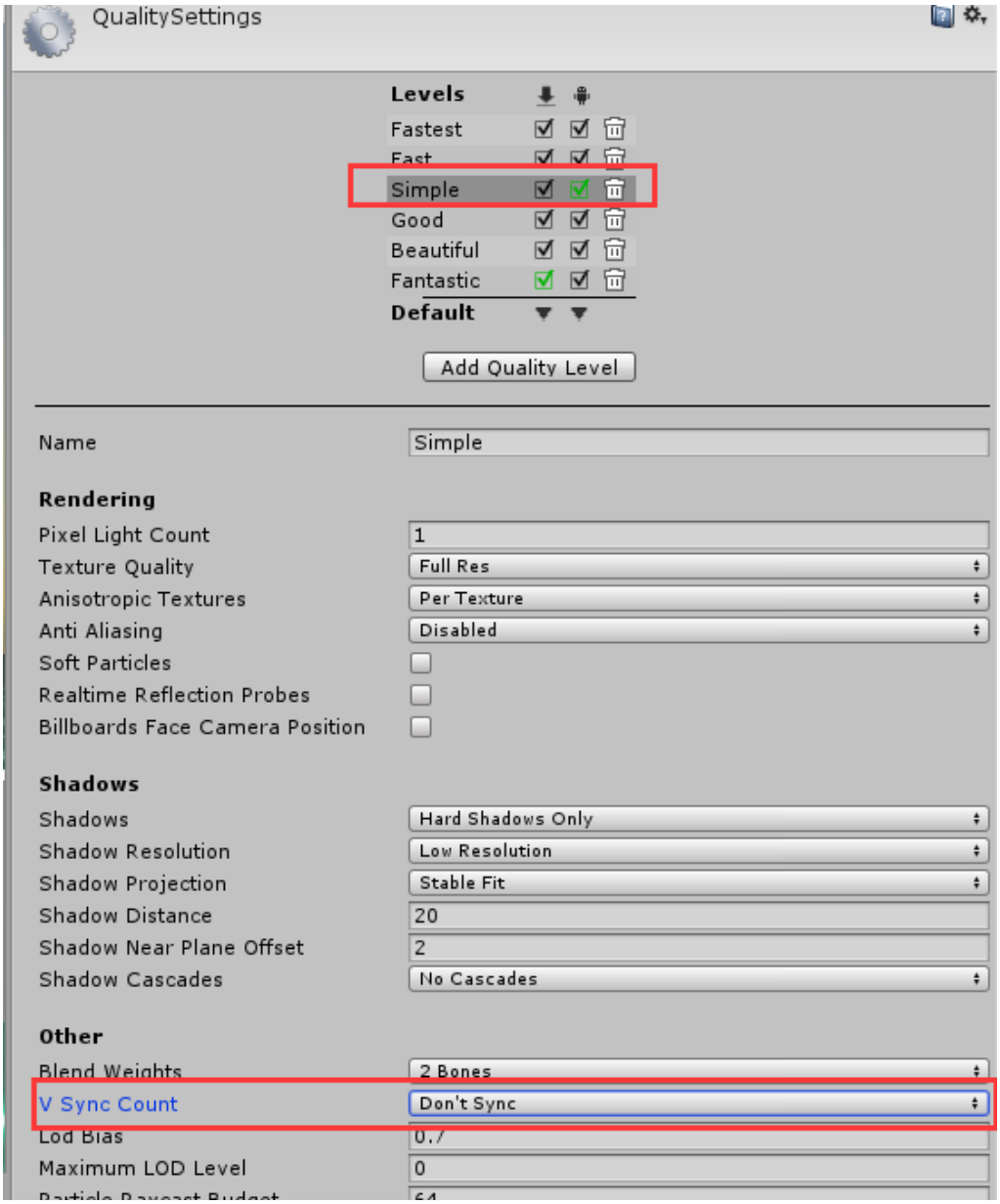


Figure3.1.2

3.2 Android Mobile Requirements

Requirement for Android phones:

Android Version	Android 4.4 and later
Mobile Sensors	Accelerator and Gyroscope

4 SDK Components

4.1 PicoVR Android SDK directory

The folders below (Figure 4.1) could be found after Pvr_Unity_SDK.unitypackage is imported successfully.

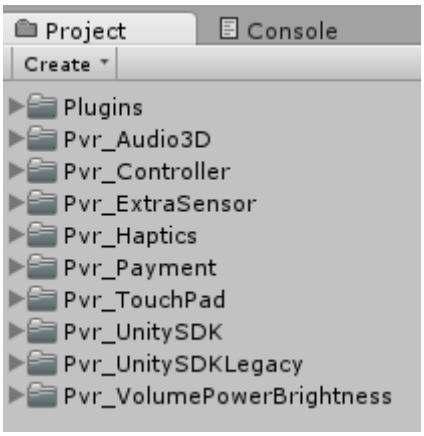


Figure4.1 SDK Folder

- Pvr_UnitySDK: The SDK for VR development，core directory，**other functions depend on Pvr_UnitySDK。**
- Pvr_Audio3: it provides scripts, prefabs, demos for 3D sound effect, depending on Pvr_UnitySDK。
- Pvr_Controller: it provides scripts, prefabs, demos for Pico Controller, depending on Pvr_UnitySDK.
- Pvr_ExtraSensor: it provides scripts, prefabs, demos for multi-sensor usage cases, depending on Pvr_UnitySDK.
- Pvr_Haptic: it provides scripts, prefabs, demos for haptic functions, depending on Pvr_UnitySDK.

- Pvr_Payment: it provides scripts, prefabs, demos for payment function, depending on Pvr_UnitySDK.
- Pvr_UnitySDKLegacy: it provides SDK compatible with old Unity version，which includes scripts, prefabs, demos, depending on Pvr_UnitySDK.
- Pvr_VolumePowerBrightness: it provides APIs for system volume, power, brightness, etc. depending on Pvr_UnitySDK。
- Plugins: the so & jar needed.

4.2 PicoVR Android SDK function description

4.2.1 Pvr_UnitySDK

The core function of PicoVR Android SDK, it can implement basic function. It is recommended to use the provided prefab, as the figure below. Be careful to modify the scripts under this directory when necessary.

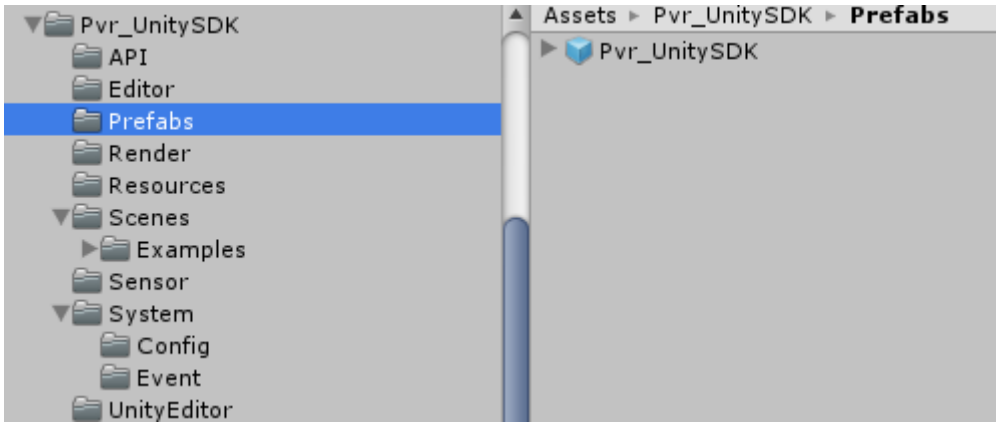


Figure 4.2 Pvr_UnitySDK directory

Pvr_UnitySDK contains APIs for sensor and rendering core functions, the detailed description please refer to Chapter 9.

4.2.2 Pvr_Audio3D

PicoVR Android SDK provides 3D sound effect implementation, the developers can use the provided API referring to Chapter 9, and also the prefab is available.

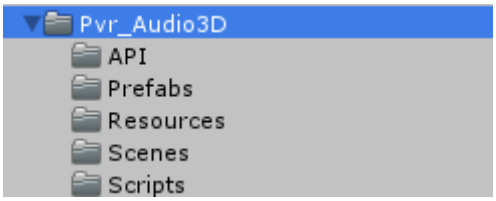


Figure 4.3 Pvr_Audio3D directory

4.2.3 Pvr_Controller

PicoVR Android SDK provides Pico Controller for development , the provided prefab is recommended. Be careful to modify the scripts under this directory when necessary. The API description can be found in Chapter 9.

4.2.4 Pvr_ExtraSensor

PicoVR Android SDK provided function to develop with multiple sensors; developers can add the provided script (Figure4.4) onto the game object (Figure 4.5).

Figure 4.4 Pvr_ExtraSensor Script

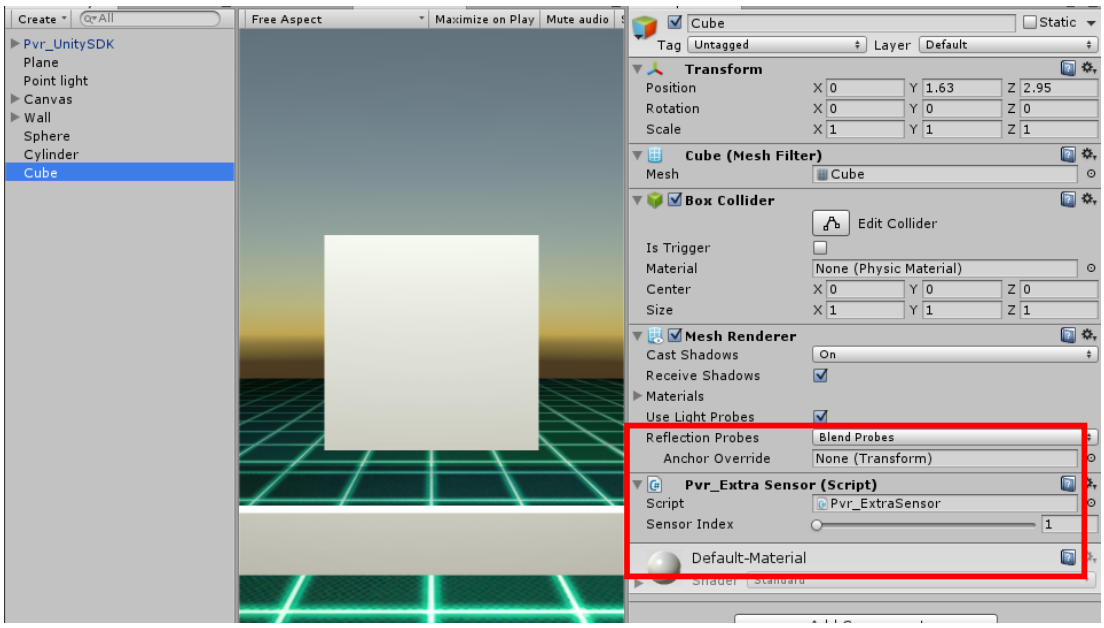


Figure 4.5 Multiple sensor

4.2.5 Pvr_Haptics

PicoVR Android SDK provides haptic function only for Pico Neo. The detailed description can be found in Chapter 9.

4.2.6 Pvr_Payment

The detailed description can be found in Chapter 8.

4.2.7 Pvr_UnitySDKLegacy

The SDK compatible with old version Unity, it is recommended to use new SDK instead. The detailed introduction of Legacy SDK can be found in the original manual.

4.2.8 Pvr_VolumePowerBrightness

PicoVR Android SDK provides APIs for system volume, power and brightness,

some API can only support for Pico Neo.

4.2.9 Pvr_TouchPad

The PicoVR Android SDK provides a call to Pico Lark, Pico Lark1 touch pad. See section 6.3 for specific use.

5 PicoVRSDK Development Process Guidance

5.1 Pvr_UnitySDK Development Guidance

5.1.1 **Create Project**

- Click "File" menu

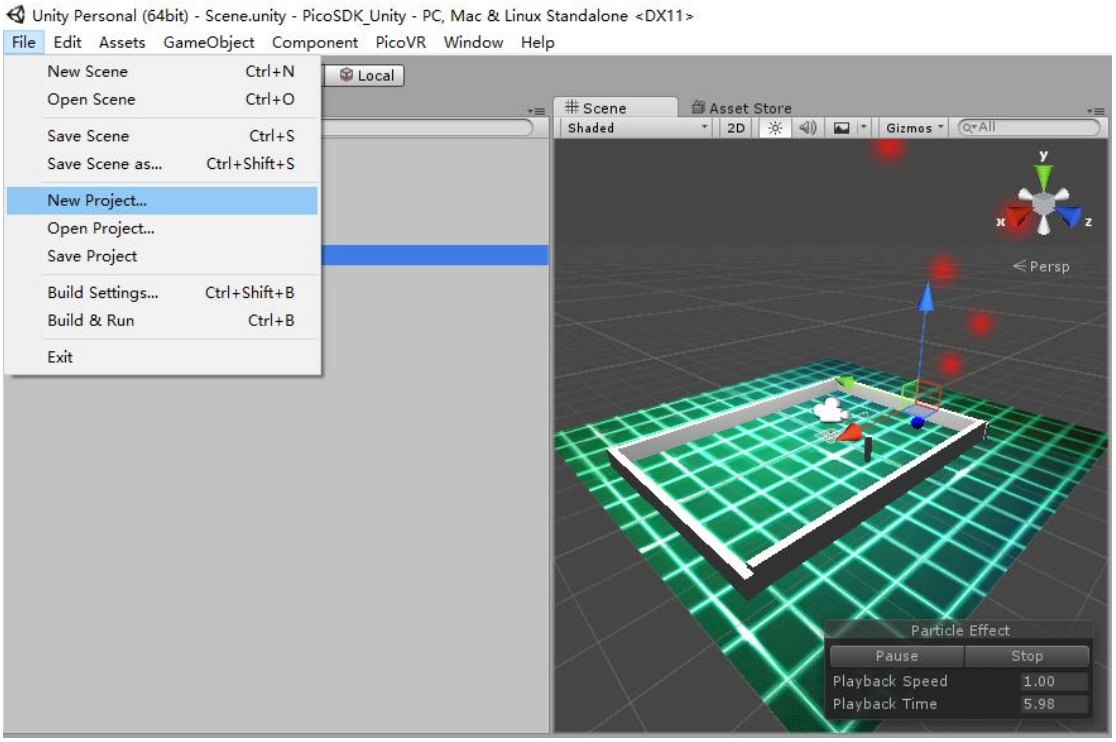


Figure5.1 File Menu

- Select "New Project"

- Create a new project according to Unity guidance

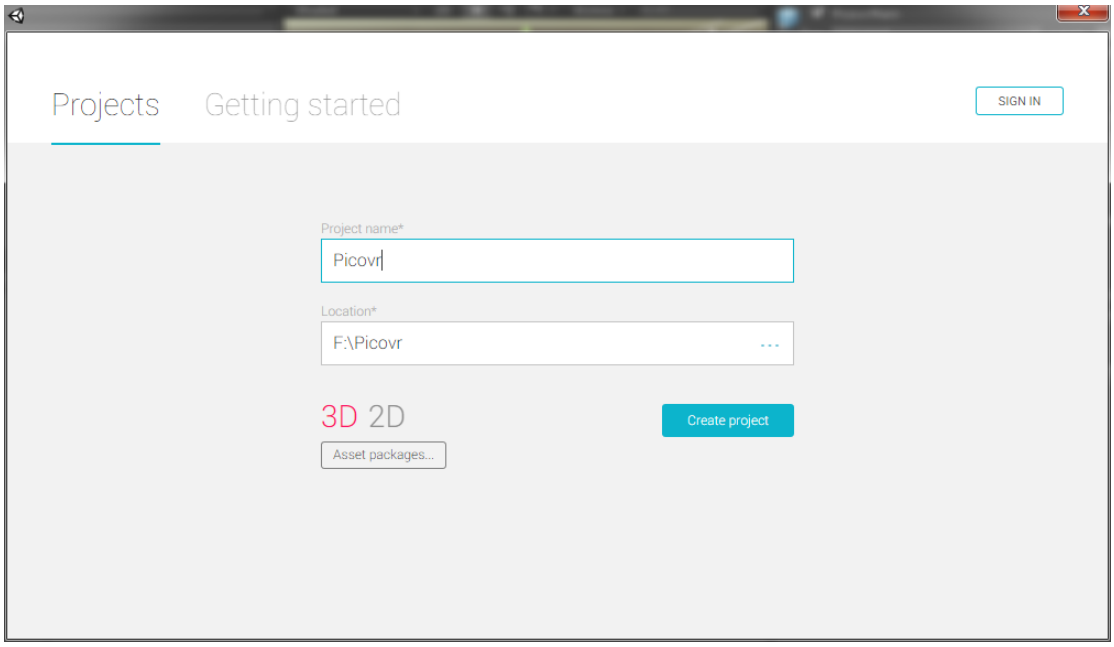


Figure5.2 Create Unity Project

5.1.2 Import Development Package

- Click "Assets" menu
- Select "Import Package" menu > Select "Custom Package"

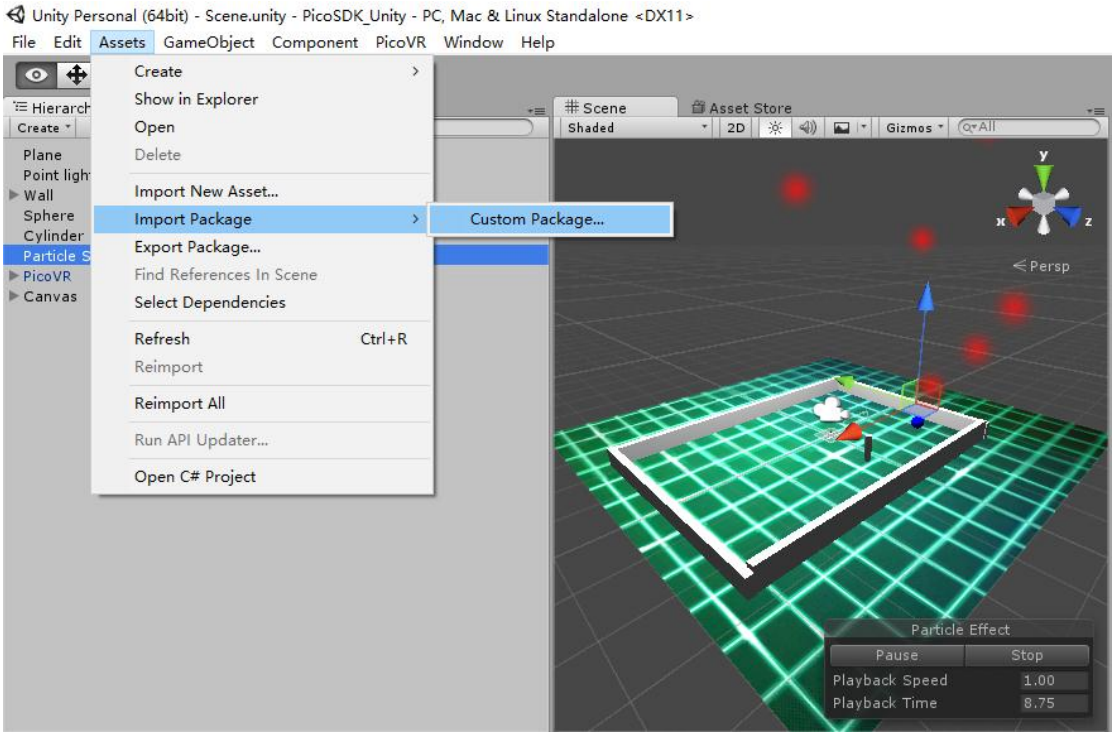


Figure5.3 Import Package

- Browse to Pvr_UnitySDK.unitypackage
- Click “Import”, and import all the content

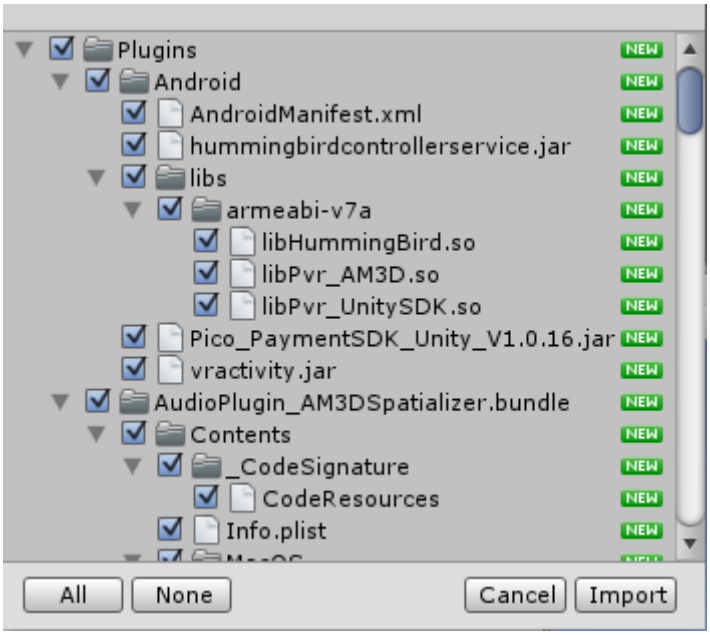


Figure 5.4 Import Package

5.1.3 Replace Camera

- Delete "Main Camera" that comes with the scene
- Find the "Pvr_UnitySDK\Prefabs" folder in the Project View

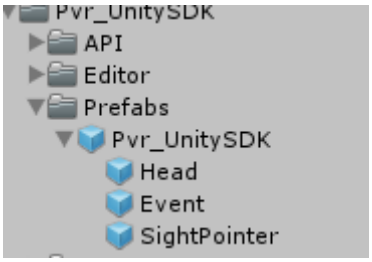


Figure 5.5 Prefabs Folder

- Drag Pvr_UnitySDK.prefab to the scene
- Set position to "0,0,0" (This is for test, and the developer can drag the prefab to the desired scene position)

5.1.4 Set Scenes

- Create Cube1，set the position information as follows

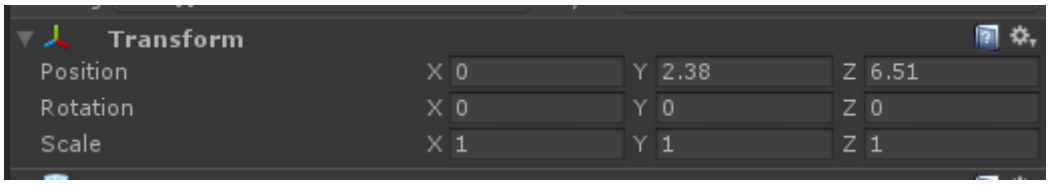


Figure 5.6 Cube1 Attributes

- Create Cube2，set the position information as follows

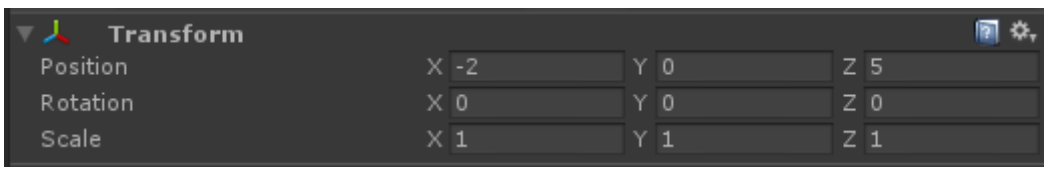
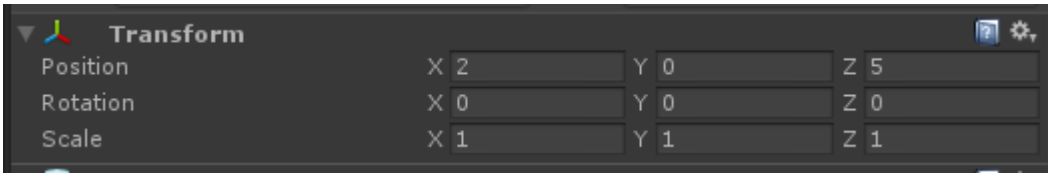


Figure 5.7 Cube2 Attributes

- Create Cube3，set the position information as follows:



➢ Figure 5.8 Cube3 Attributes

➢ After completing the configuration, the hierarchy in the scene is shown in Figure 5.9.

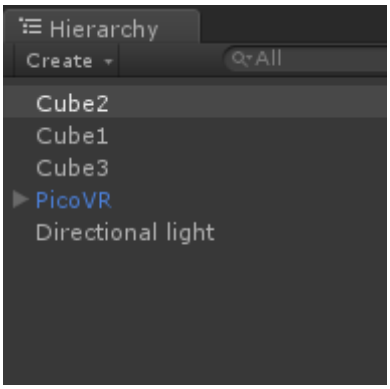


Figure 5.9 Scene Hierarchy

5.1.5 Simulation Run

➢ Click "Play" in the editor, and the results are as follows:

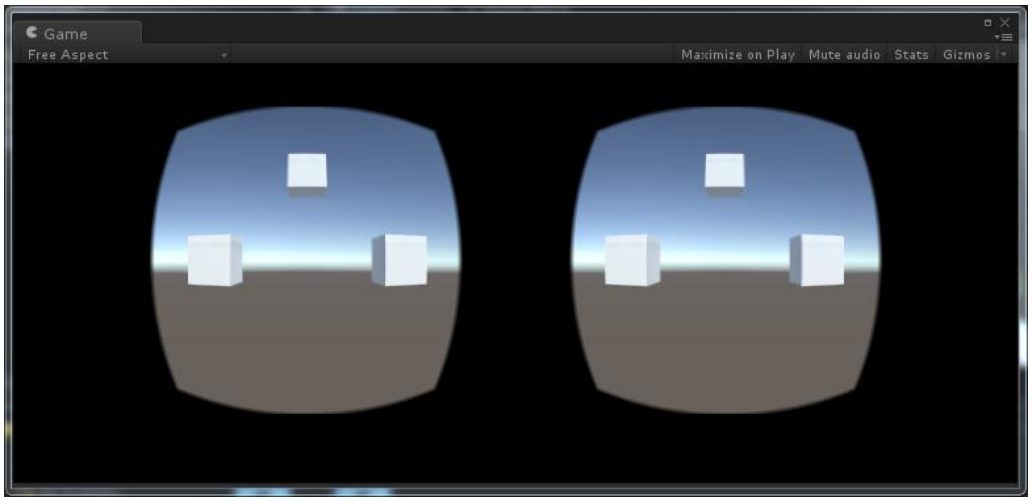


Figure 5.10 Run Example

5.1.6 Simulation Operation

- Hold Alt, and move the mouse, and then the screen will rotate according to the operation.
- Hold Alt and click the left mouse button to select VR mode or Mono mode (Unity Editor only)

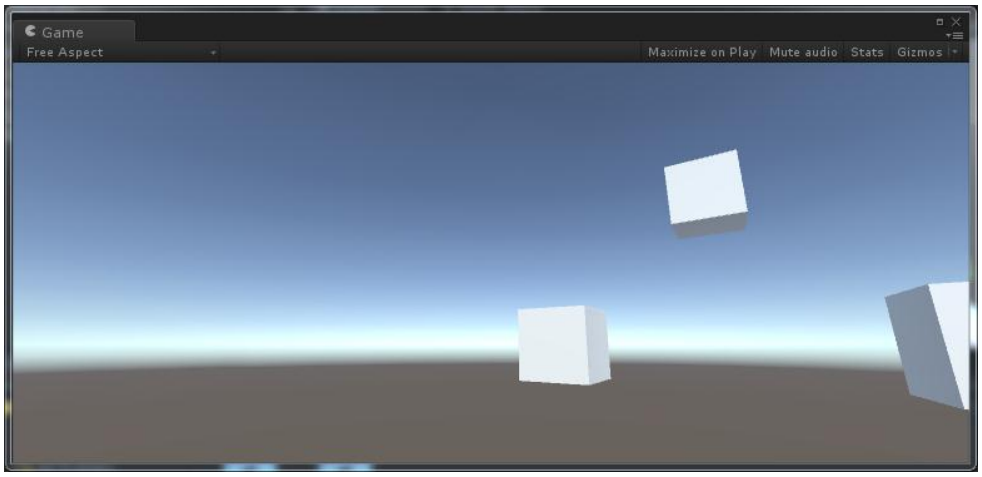
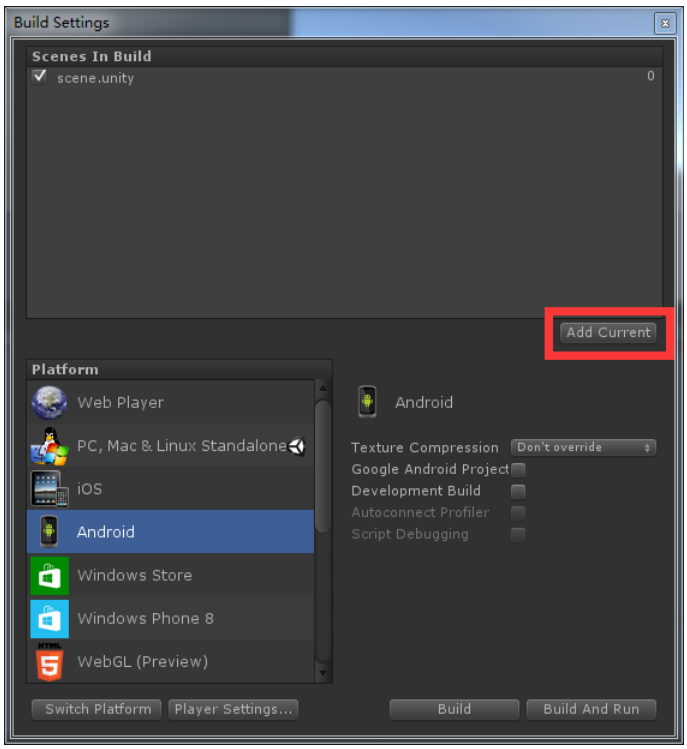


Figure 5.11 Mono Mode Example Diagram (Unity Editor Only)

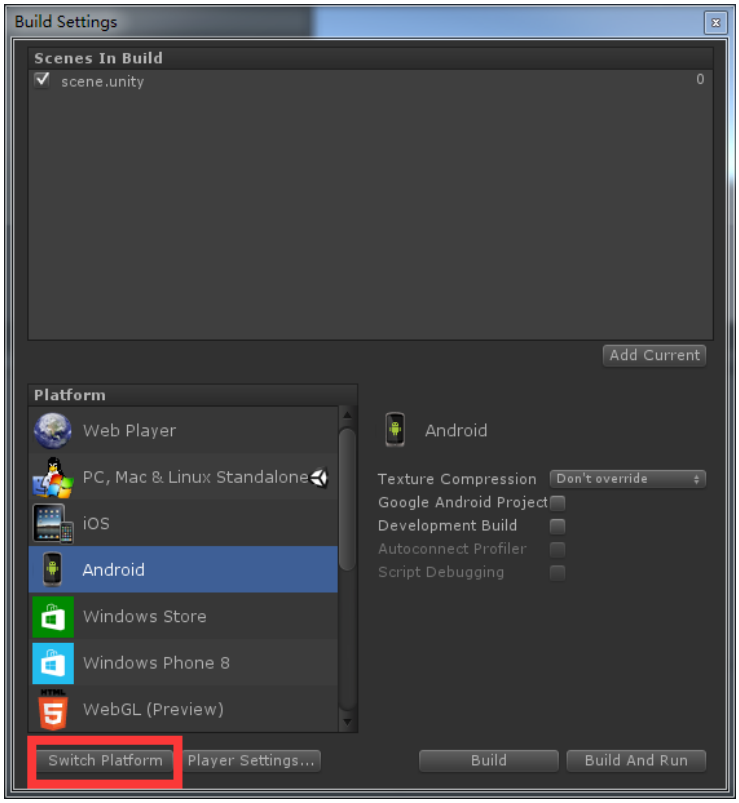
5.1.7 Publish Apps to Android

- Firstly, add scenes in the Build Settings



➢ Figure 5.12 Add Scenes

➢ Select Android in the Platform option, and click Switch Platform



➢ Figure 5.13 Switch Platform

- Click Build，and then Android application packaging will succeed

5.1.8 About AndroidManifest

- **Case 1: AndroidManifest file is not included in the application**

The AndroidManifest file in the SDK file can be directly used.

- **Case 2: The AndroidManifest file is existed in the application**

  Be sure to merge the AndroidManifest files from SDK with original one.

  Mainly contains the following points:

  A. In Pico Neo project, you need to add a special meta-data, otherwise exception will be displayed in the Pico Neo

```
<meta-data android:name="com.picovr.type" android:value="vr"/>
<meta-data android:name="com.picovr.display.orientation"
android:value="180"/>
```

  B. In Pico Neo project, all Activitys need to be displayed should inherit from the com.Unity3d.player.UnityPlayerNativeActivityPico

  C. In Pico 1 or Pico 1s project, the following Service definitions should be added:

```
<service android:name="com.picovr.picovrlib.service.LarkConnectService"
/>
<service android:name="com.picovr.picovrlib.ble.BluetoothLeService" />
```

  D. Add necessary permissions

```
<uses-permission android:name="android.permission.WRITE_SETTINGS"/>
<uses-permission android:name="android.permission.BLUETOOTH" />
<uses-permission android:name="android.permission.BLUETOOTH_ADMIN" />
<uses-permission android:name="android.permission.INJECT_EVENTS" />
```

6 Operations of Controller

6.1 Pico Controller

1. Put the Pvr_Controller prefab under Pvr_UnitySDK/Head

Figure6.1.1

As below

Figure6.1.2

2. Pvr_Controller Inspector

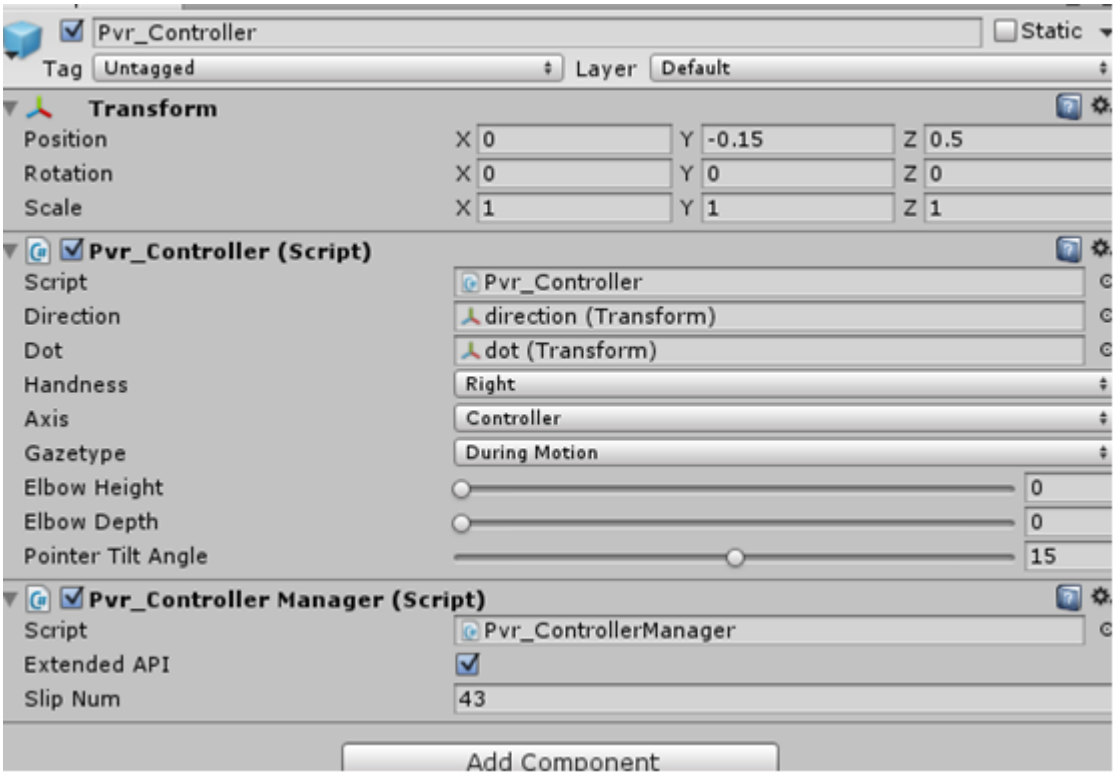
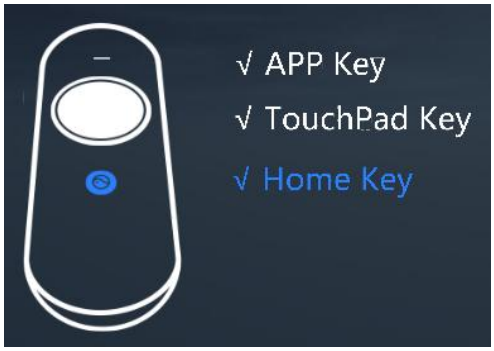


Figure6.1.3

3.Pico Controller key description

Corresponds to APPKey in API

Corresponds to TouchPadKey in API



Corresponds to HomeKey in API

Corresponds to VolumeUpKey in API

Corresponds to VolumeDownKey in API

Figure6.1.4

4.Pico Controller TouchPad description

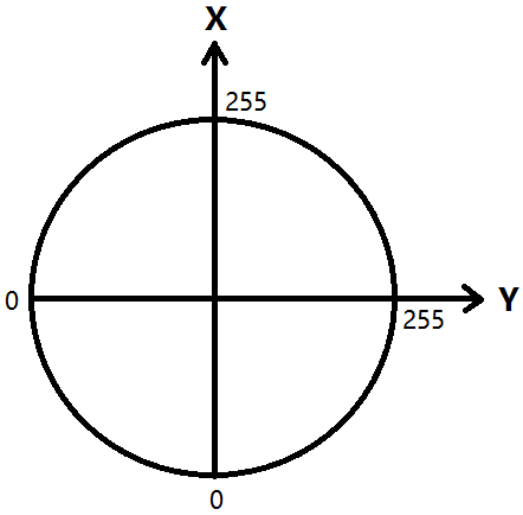


Figure6.1.5

The slide function of the Hummingbird handle is the lifting of the finger to determine the end of the slide.

The related API of Pico Controller can be found in Chapter 9.2.4.

5.Handle Controller automatic connection function.(Common Phone)

Open by default,After the Pvr_Controller Demo running,it will automatically search for the nearby handle controller, and automatically connect the first search to the handle controller, when the connection is successful,it will stop looking for. If the connection is not successful, will popup tooltip after 15 seconds, prompt the user

to search or exit.

If you need to turn off the automatic connection function. You need to modify the bool isAutoConnect in the Pvr_ControllerLink.cs file to false. .

6.2 Pose Tracking of Pico Neo Controller

Figure6.2.1 Pico Neo Controller

Compatible with Pvr_UnitySDKLegacy version, Activate the sensor tracking of Pico Neo Controller by checking the switch as Figure6.2.2

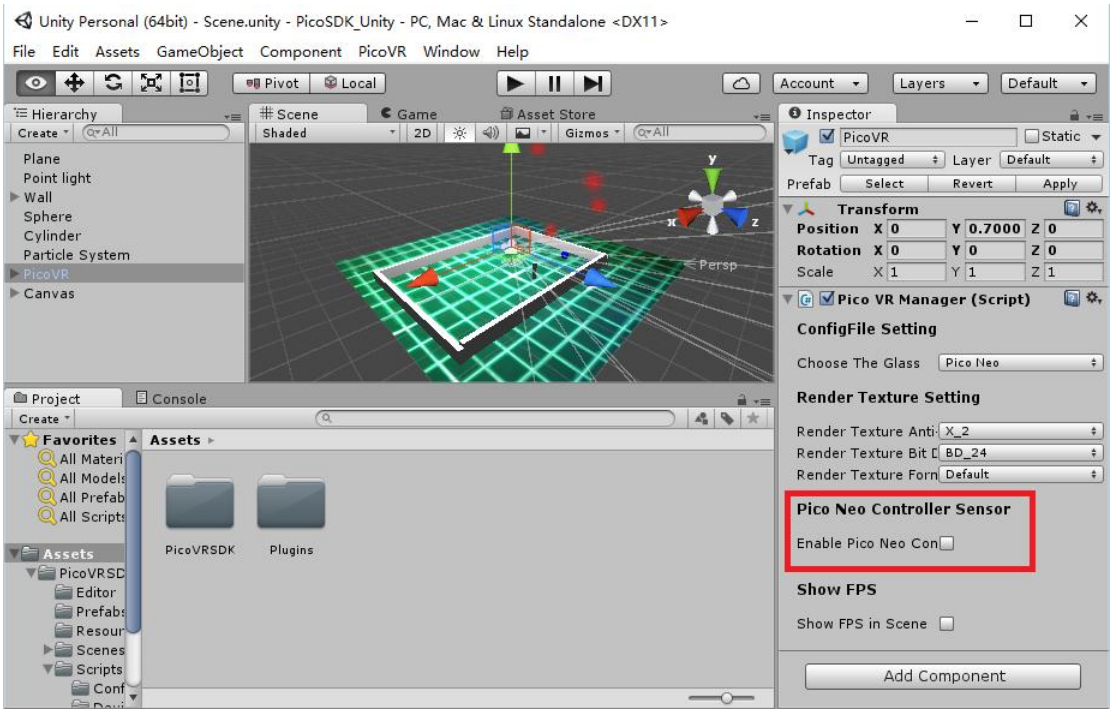


Figure 6.2.2 Pico Neo Controller Sensor Switch

Once activated, the variable PicoVRManager.SDK.boxQuaternion will be updated in real time. You can get the Quaternion of the controller via the following interface: public Quaternion getBoxQuaternion().The Quaternion also can be get by reading the boxQuaternion variable directly.

In 4.2.4 to konw the new Way using pico neo controller in new version

6.3 Touch Panel Operation Settings of Pico 1 and Pico 1s

Compatible with Pvr_UnitySDKLegacy version

- Use startLarkConnectService & stopLarkConnectService to start or stop the local service as Figure 6.3: (Need pass Unity GameObject name as a parameter when use startLarkConnectService for callback. The callback is used in BLE service, and it will not be called in SPP service)
- SPP service will be started according to the parameters of Pico 1 set by the PicoVR application , while BLE service will be started according to the parameters of Pico 1s. BLE service will be started by default instead of SPP

service if no parameters provided. So, please install the PicoVR application first, if you want to connect to Pico 1.

- Choose to use Pico 1 or Pico 1s in application "PicoPlayer", SPP service or BLE service will be started by calling the method of startLarkConnectService.

```
UnityEngine.AndroidJavaClass unityPlayer = new UnityEngine.AndroidJavaClass("com.unity3d.player.UnityPlayer"
activity = unityPlayer.GetStatic<UnityEngine.AndroidJavaObject>("currentActivity");
javaVrActivityClass = new UnityEngine.AndroidJavaClass("com.picovr.picovrlib.VrActivity");

public void startLarkConnectService()
{
    Debug.LogError("shine startLarkConnectService ");
    CallStaticMethod(javaVrActivityClass, "startLarkConnectService", activity, PicoVRManager.SDK.gameObject.name);
}

 public void stopLarkConnectService()
 {
     Debug.LogError("shine stopLarkConnectService ");
     CallStaticMethod(javaVrActivityClass, "stopLarkConnectService", activity);
 }

public bool CallStaticMethod(UnityEngine.AndroidJavaObject jobj, string name, params object[] args)
{
    try
    {
        jobj.CallStatic(name, args);
        return true;
    }
    catch (AndroidJavaException e)
    {
        Debug.LogError("CallStaticMethod  Exception calling activity method " + name + ": " + e);
        return false;
    }
}
```

Figure 6.3 Start/Stop SPP Service

- Handle the callback BLEStatusCallback as Figture 6.4:

```
public const int SERVICE_STARTED = 0;
public const int CONNECTE_SUCCESS = 1;
public const int DISCONNECTE = 2;
public const int CONNECTE_FAILED = 3;
public const int NO_DEVICE = 4;
public void BLEStatusCallback(string s)
{
    switch(int.Parse(s)){
    case(SERVICE_STARTED):
        Debug.Log("BLE_SERVICE_STARTED");
        break;
    case(CONNECTE_SUCCESS):
        Debug.Log("BLE_CONNECTE_SUCCESS");
        break;
    case(DISCONNECTE):
        Debug.Log("BLE_DISCONNECTE");
        break;
    case(CONNECTE_FAILED):
        Debug.Log("BLE_CONNECTE_FAILED");
        break;
    case(NO_DEVICE):
        Debug.Log("BLE_NO_DEVICE");
        break;
    }
}
```

Figure 6.4 BLEStatusCallback

New Way to use pico1 and pico1s in new version

Use reference Pvr_TouchPad/Script/Pvr_TouchPad.cs specific use can change Pvr_TouchPad/Script/Pvr_TouchPad.cs in the Update () method, complete the relevant operation.

Handle your own business according to the key codes

(effect in both two versions)

Pico Action	**Android KeyEvent**	**Unity Keycode**
TOUCH_PAD _CLICK	KEYCODE_BUTTON_A	Joystick1Button0
TOUCH_PAD _UP	KEYCODE_DPAD_UP	UpArrow

TOUCH_PAD _DOWN	KEYCODE_DPAD_DOWN	DownArrow
TOUCH_PAD _LEFT	KEYCODE_DPAD_LEFT	LeftArrow
TOUCH_PAD _RIGHT	KEYCODE_DPAD_RIGHT	RightArrow
ACTION_PICO_BACK	KEYCODE_BACK	Escape
ACTION_PICO_MENU	KEYCODE_MENU	Menu
ACTION_VOLUME_CHANGE	Intent Broadcast	--
ACTION_CAMERA	KEYCODE_NUMPAD_2	Keypad2
ACTION_AUDIOJACK_IN	KEYCODE_NUMPAD_3	Keypad3
ACTION_AUDIOJACK_LOSE	KEYCODE_NUMPAD_4	Keypad4
ACTION_SENSOR_NEAR	KEYCODE_NUMPAD_7	Keypad7
ACTION_SENSOR_FAR	KEYCODE_NUMPAD_8	Keypad8

- ACTION_VOLUME_CHANGE uses broadcast mechanism, please receive and process the broadcast as below:

  Action：com.picovr.picovrlib.service.volume

  Extra： hfpVolume (int) & a2dpVolume (int)

7 Debugging Tool

7.1 Usage of FPS Displaying

FPS displaying is only for development and testing, and please see the comments in FPS_S.cs for details. FPS displaying is set to off by default, please check the

"Show FPS in Scene" to turn it on.

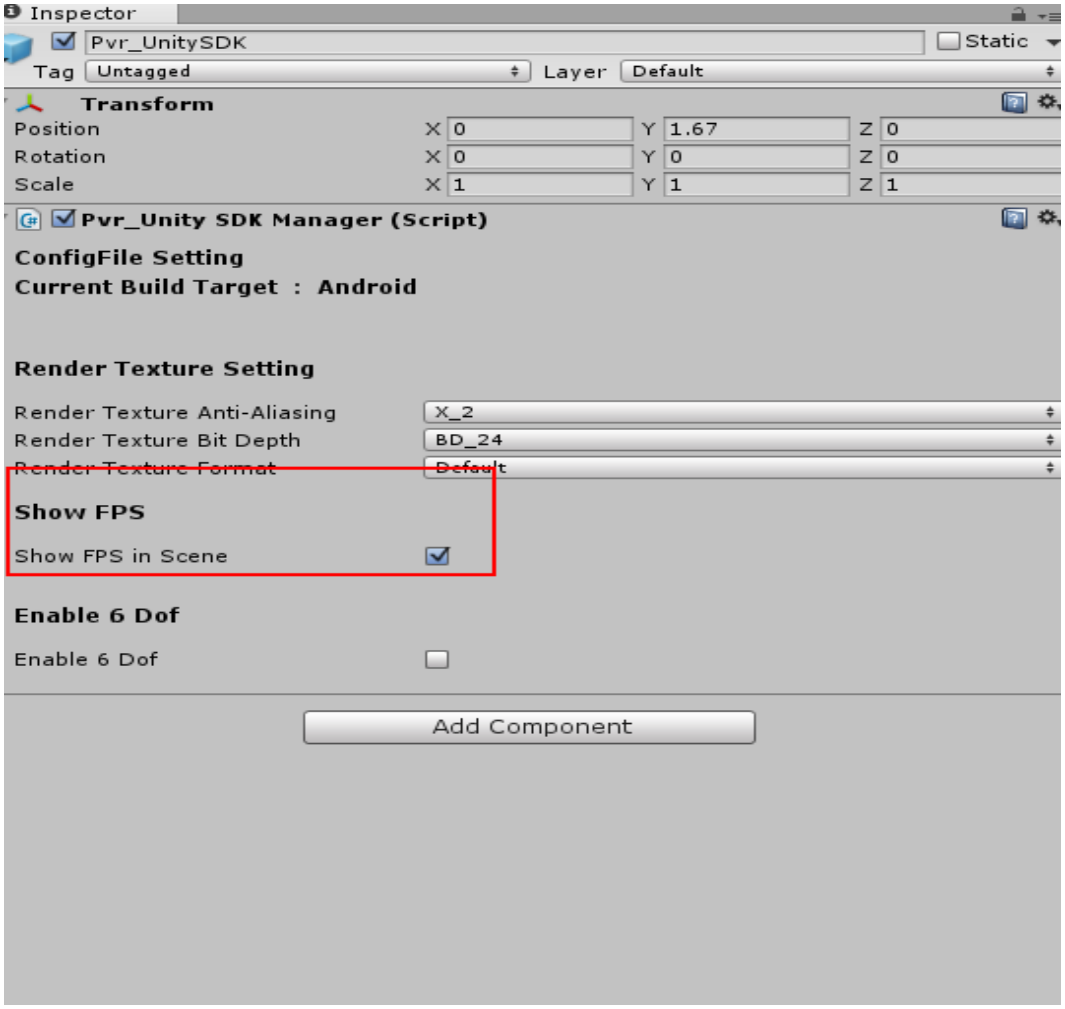
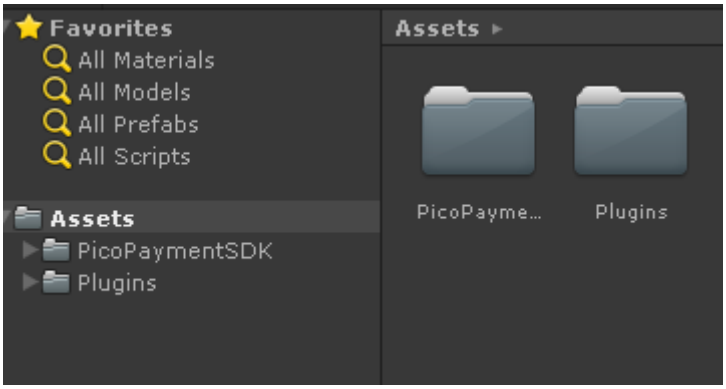


Figure7.1 Turn On FPS Display

8 Payment SDK Instructions(Pico Neo Only)

8.1 SDK package

8.1.1 Pico Payment SDK Directory

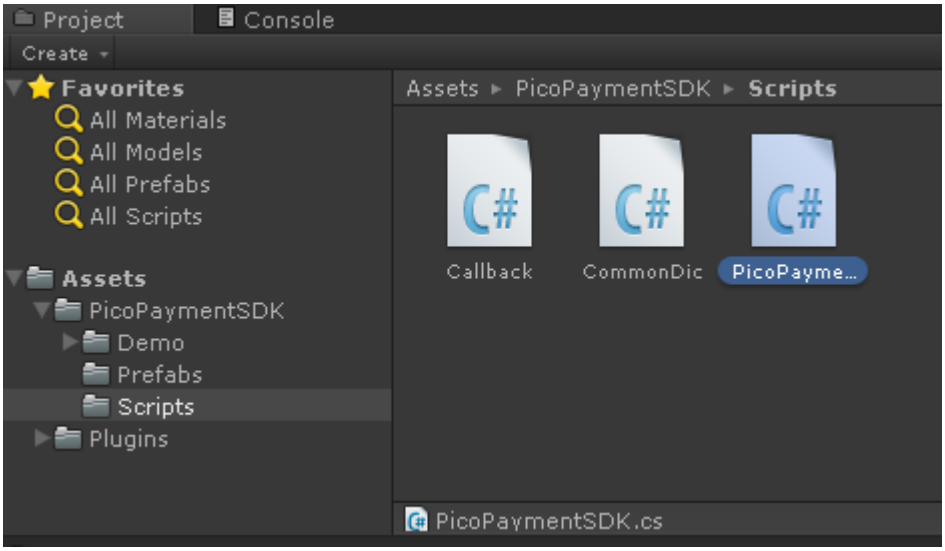


"PicoPaymentSDK" includes the source code of the core functions of this SDK and a Demo for showing the functions.

"Plugins" includes .jar packages as well as .so files that requires.

8.1.2 Pico Payment SDK Content

- Scripts

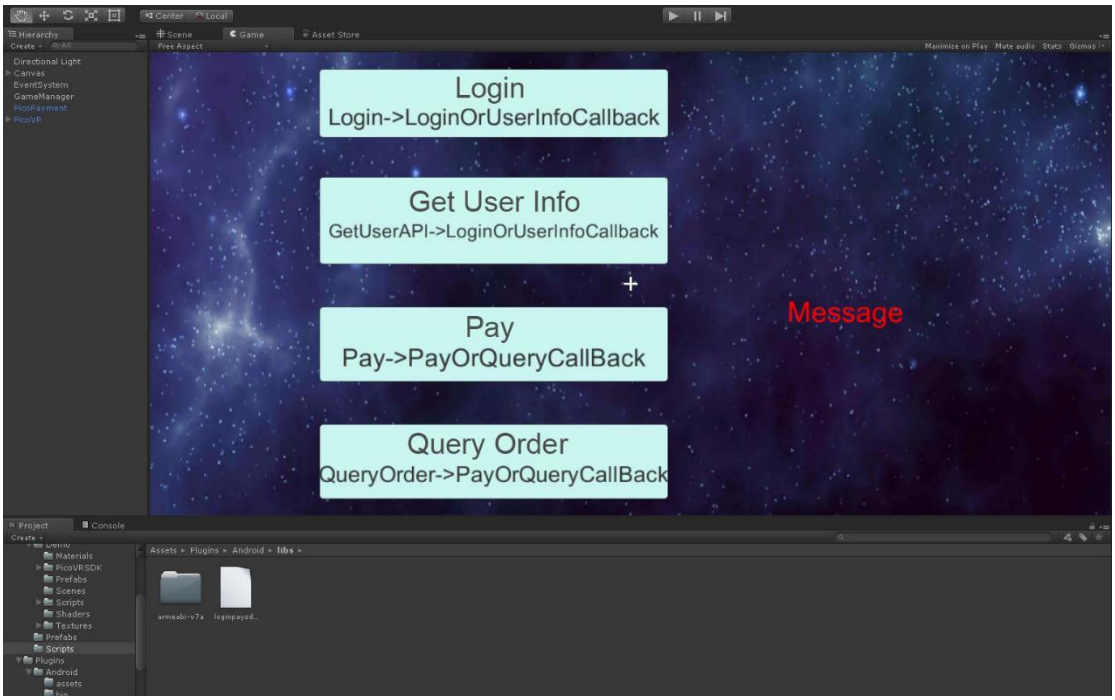


Developers only need to call methods in PicoPaymentSDK.

Including: Login (), Pay (string payOrderJson), QueryOrder (string orderId) and GetUserAPI (). For more details, please refer to 9.3.

- Demo

Scene that showing SDK functions as following picture needs to be used with PicoVRSDK.



As you can see, the left part of each button is the method needs to be called, and the right part is the callback method.

About Callback

1、You can see a prefab named "PicoPayment" under Prefabs folder. Please put this prefab into your project since it is for callback for the system.

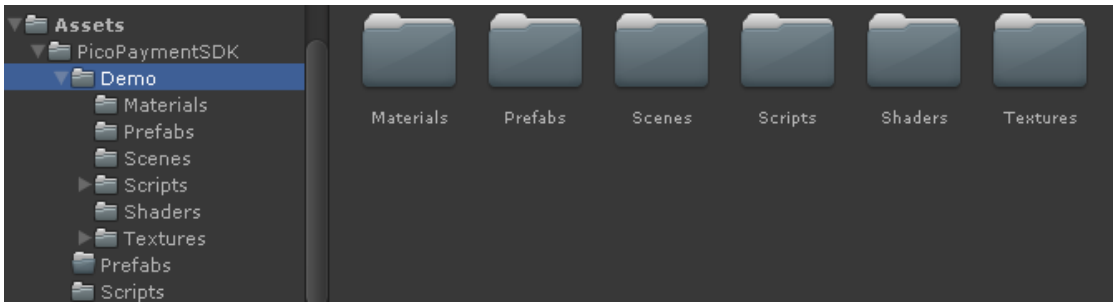
2、The "Callback" script attached to "PicoPayment" prefab preprocessed part of data of callback. Please feel free to modify them except method names if developers need to process other data.

8.2 Pico Payment SDK Using Guidance

8.2.1 Understand Functions through The Build-in Demo of SDK

Double click Demo – Scenes – Demo.

Notice: Demo requires PicoVRSDK to use. Debug environment is PicoNeo with PUI version above 1.3.2.



8.2.2 About AndroidManifest

This SDK implemented AndroidManifest file that you can use directly if your project does not include any AndroidManifest files.

Otherwise, you will have to merge the content as followings to your AndroidManifest:

- Required permission:

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

- Activity declaration:

```
<activity android:name="com.pico.loginpaysdk.UnityAuthInterface">
<intent-filter>
<action android:name="android.intent.action.MAIN"/>
<category android:name="android.intent.category.LAUNCHER"/>
</intent-filter>
</activity>
<activity android:name="com.pico.loginpaysdk.component.PicoSDKBrowser"
android:configChanges="keyboardHidden|orientation"
android:windowSoftInputMode="adjustResize"
android:exported="false">
</activity>
```

- Developer information:

```
<meta-data
android:name="pico_merchant_id" android:value="company_id"/>
<meta-data android:name="pico_app_id"
android:value="1d6ef7f25a7b0ec3bbd5b6bf247adf71"/>
<meta-data
android:name="pico_app_key"android:value="2890d4a291108e73ef0e87340affe7a4"/>
<meta-data
android:name="pico_pay_key" android:value="pico2016"/>
```

From above parameters, pico_app_id and pico_app_key needs to be filled strings that acquired from official website. pico_merchant_id and pico_pay_key can be above values before you can acquire from official website.

Tip : In this configuration file, pico_pay_key corresponds to the AppSecret on the developer platform.

8.2.3Obtain AppKey, AppID and AppSecret method

The current developer platform support PicoStore, PicoHome, PicoVR application release and management functions.Pay for the SDK developers to access, need to create applications and obtain the corresponding string on the developer platform.Application process is as follows:

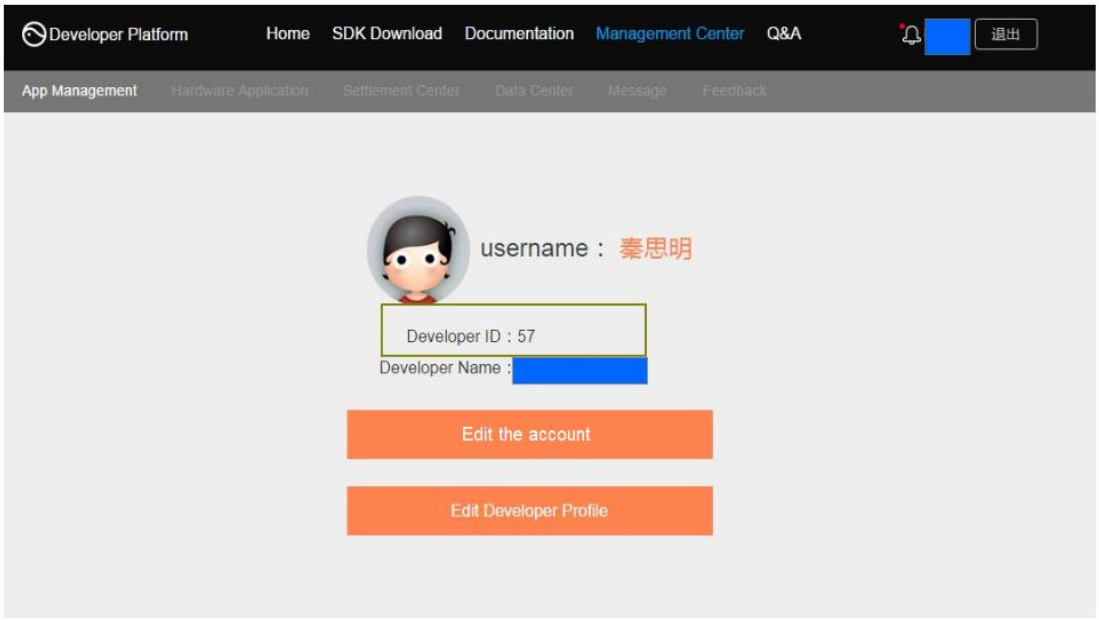
1. Login developer platform and register Pico member (http://us-dev.picovr.com/)

   Before application developers need to become a member of Pico.

2. Apply for developers

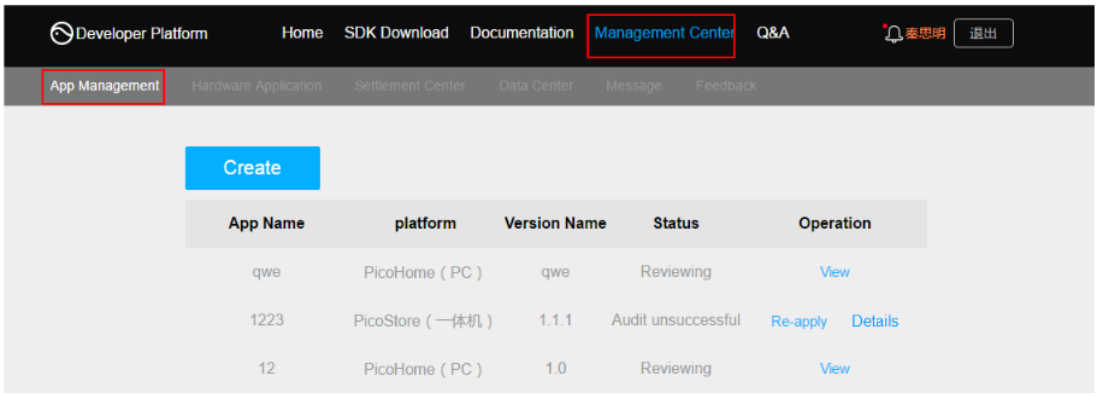
   Developer is divided into individual developers and companies, please according to the actual situation of the application.Review is submitted, we will feedback within 3 working days, please check the developer platform.

3. Check the merchant ID



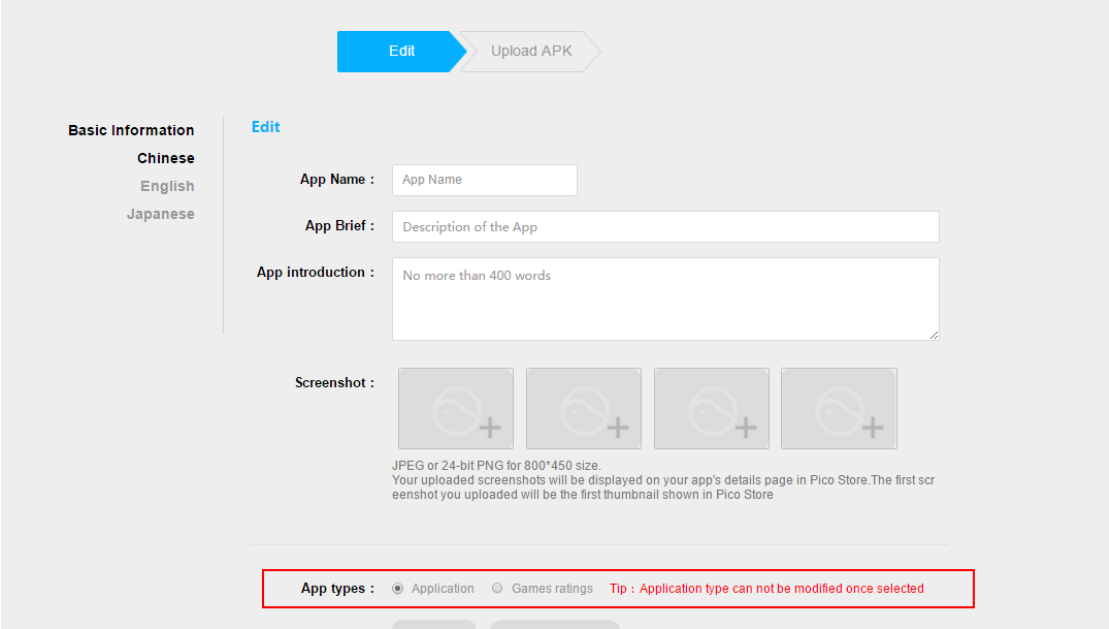
After apply for developers, click on the upper right corner of the nickname can view to the developer ID, merchant ID.

4.Create Application

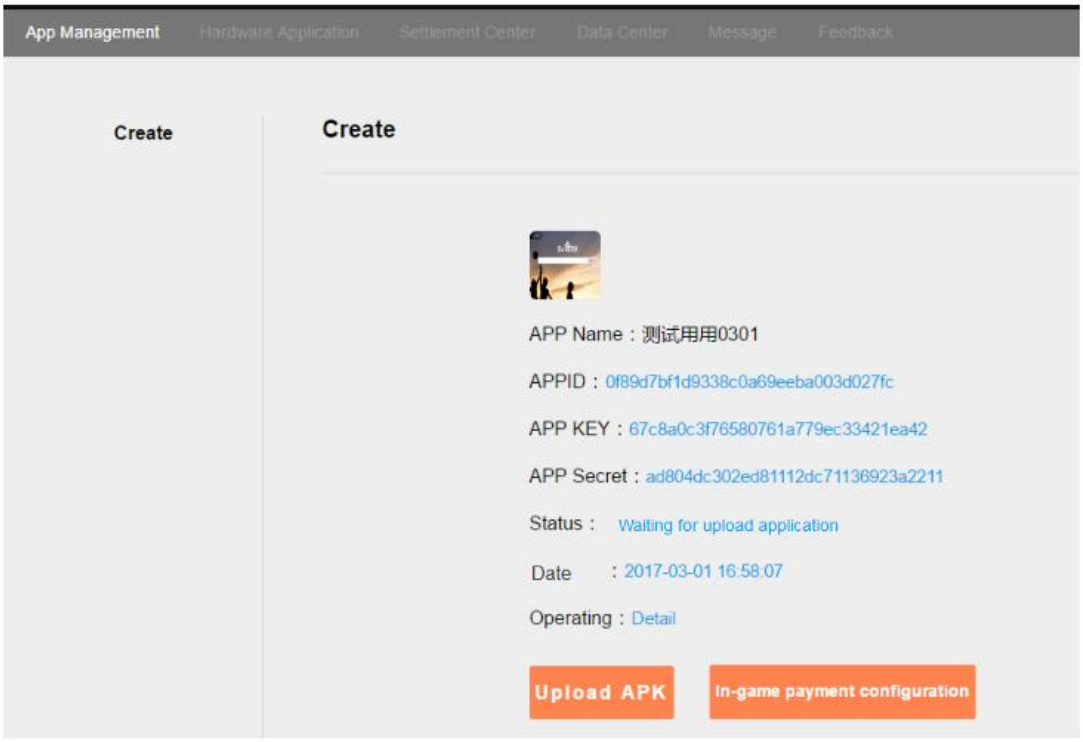


Developers can entered the stage of creating applications from the management center.By creating applications access to AppKey, AppID and AppSecret string.

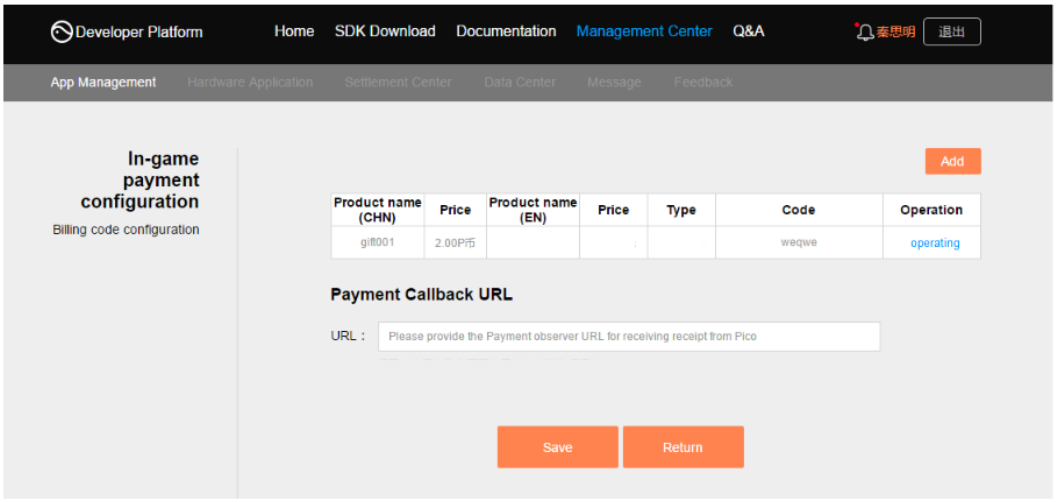
After creating an, can choose to release platform



After entering the corresponding platform can improve the application of information related to specific rules (see the developer front page).Please focus attention on the red icon position, please carefully fill in the application type, once complete cannot be modified!Games application if there are items in pay, we require the developer must adopt the background increase commodity code in a way that unified management.



Successfully create application, developer platform will be allocated string to it。



8.3 Pico Payment SDK Function Interface Instructions

8.3.1 Login

- Method to call
  - Main method to call：void Login()

- Explanation: needs to call this method when login.

➢ Callback method

- void LoginOrUserInfoCallback(string LoginOrUserInfo)
- Return values explanation: LoginOrUserInfo // return login information from backend. If login successfully, the format will be like followings:

  {"access_token":"25ba00fb73343ff1ec32e1c152fff291",

  "open_id":"2890d4a291108e73ef0e87340affe7a4",

  "refresh_token":"5a189befeb3b33f7df101fbecffe4f98",

  "expires_in":"1d6ef7f25a7b0ec3bbd5b6bf247adf71"}
- When login failed:

  {"exception":"AccessToken is null"} or {"exception":"auth code of response is null}
- When login canceled:

  {"cancel":"cancel"}

8.3.2 Payment

➢ Method to call

- void Pay(string payOrderJson)
- Input parameters: payOrderJson //including Json string of the order information
- Notice: payOrderJson needs to include following information(key):

subject	Order title

body	Order description
order_id	Order ID
total_fee	Total price
goods_tag	Item’s tag
notify_url	Notify URL
pay_code	product Code

Example 1 : Pay("{'subject':'Games',' body ':'Purchase entire game','order_id ':'10000','total':'10','goods_tag ':'game' }")

Example 2 : Pay("{'subject':'Games',' body ':' Purchase entire game','order_id ':'10000',','goods_tag ':'game','pay_code':'123' }")

Notice: Example 1 is the way to pay directly. On the other side, example 2 is the way to pay with item code. These two ways cannot process together. When paying directly, the attribute of item code cannot be filled in. Similarly, when using item code to pay, 'total' cannot be set up to 0 or anything else. Besides, these two ways need to be consistent with payment ways set up in the developers platform account.

- Callback method
  - void QueryOrPayCallback(string queryOrPayInfo)
  - Return values explanation: queryOrPayInfo // return payment information from backend. If payment successfully, the format will be like followings: {"code ":"12000","msg":"paid successfully"}

- Payment callback code&msg overview:

Return code ID and its description:

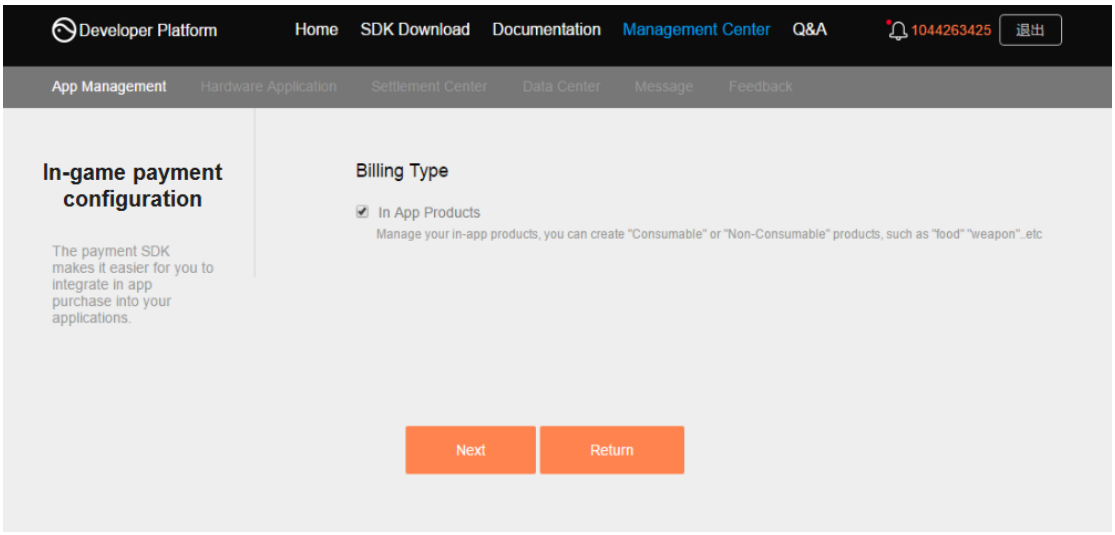
Code	Msg
00000	Network anomaly
10000	Login successfully
10001	User not login
10002	Please input correct amount
10003	Login overdue, please login again
11000	Merchant verify successfully
11001	Merchant verify failed
11002	User parameter verify failed or request overdue
11003	Merchant not verify
12000	Pay successfully
12001	Pay failed
12003	Not enough P coins
12004	Balance available
13000	Generate order
13001	Acquire data failed
13002	Generate order failed
14000	Order queried successful
14001	Order not exist or error

14002	User canceled payment action
15000	Not input item info
15001	Not input pre-pay ID
15002	Please input Pico payment order NO. or merchant order NO.
NOAUTH	Merchant not have permission for this interface
SYSTEMERROR	System error
APP_ID_NOT_EXIST	APP_ID not exits
MCHID_NOT_EXIST	MCHID not exits
APP_ID_MCHID_NOT_MATCH	app_id and mch_id not match
LACK_PARAMS	Lack of parameters
SIGNERROR	Wrong signature
NO_DATA	No data searched

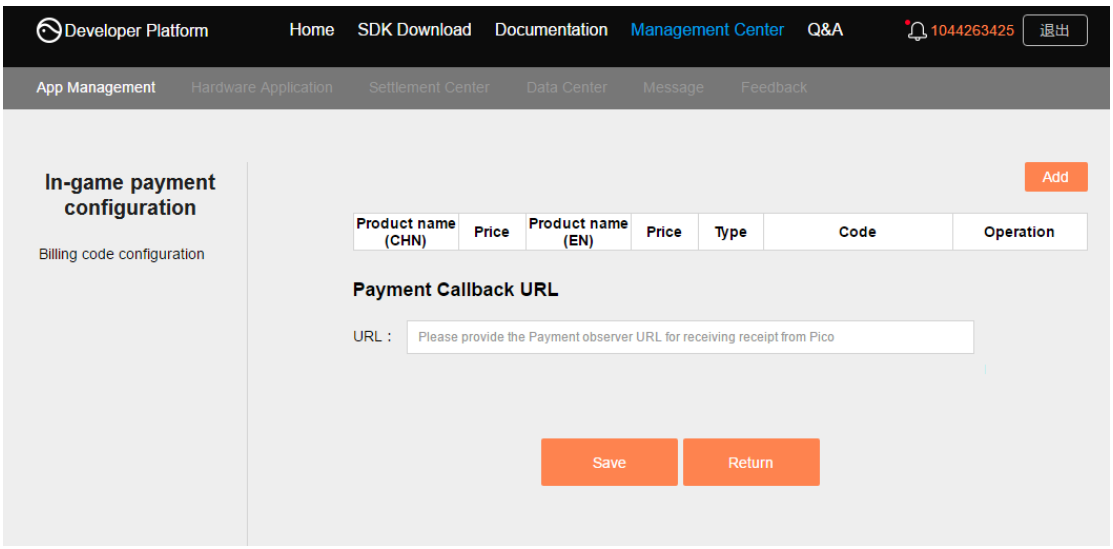
8.3.2.1 Developers pay in-game configuration instructions

Configuration premise: pay in-game configuration only according to the effect of "game type"!The choice of application type checked when creating the application, once choice cannot be modified.

Create a game after pay configuration will lead to game developers.Suggest playing before upload application configuration, reduce the audit process, the phenomenon of the leak leak test.

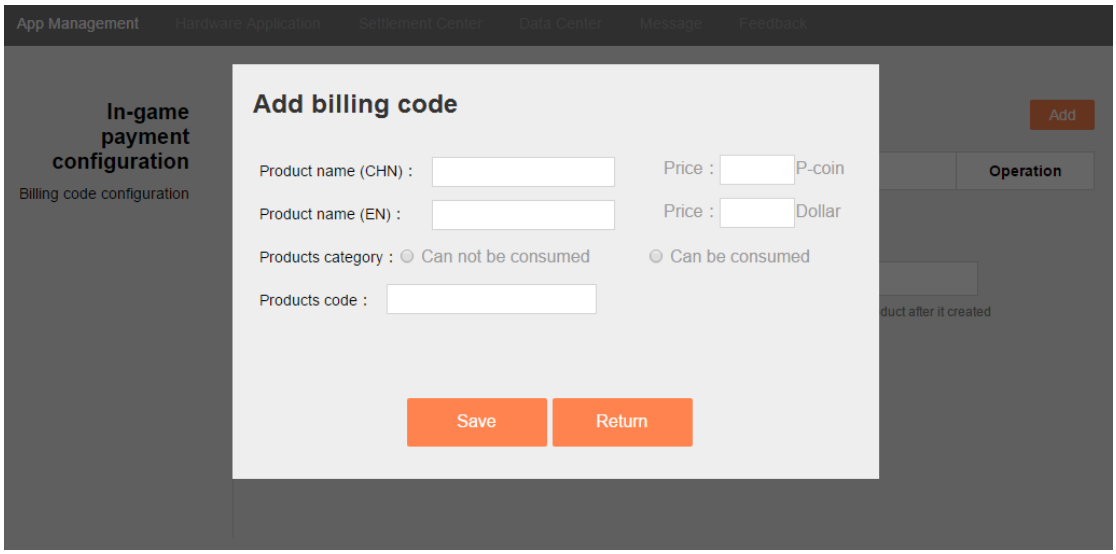


In-game configuration by commodity code configuration manner, developers need to pay within the game props accordingly configuration to take effect.



The rules of the commodity code is defined as the first letter, only allowed to enter the letters and Numbers, no more than 20 characters.The commodity code between different props can't repeat.

Item is classified into props can consume props and consumption.Can consume props for can repeat purchases, such as gold, blood, etc.;Do not consume props for one-time purchase products, such as weapons, unlock levels.



8.3.3 Query Order

- Method to call:
  - void QueryOrder(string orderId)
  - Input parameters: orderId——the order ID needs to be queried
- Callback method
  - void QueryOrPayCallback(string queryOrPayInfo)
  - Return values explanation: queryOrPayInfo——return login information from backend. It is a Json string that has not been processed.
  - Example format when queried successfully:

    {

    "trade_no":"2201608231471950587817 1324", //Pico payment order ID

    "open_id":"4f3148bdc34d9bca104927729a173b64",

    "ret_msg":"",

    "coupon_fee":0.00,

"fee_type":"PIC",
"pay_time":1471950587000, //payment completed time
"nonce_str":"yiUzuv4VQO1OXBAzVyZSRztOmRgIOioT",
"out_trade_no":"12345678903",
"trade_status":"SUCCESS", //SUCCESS—paid successfully
"trade_type":"EGG",
"result_code":"SUCCESS",
"mch_id":"company_id",
"ret_code":"SUCCESS",
"sub_msg":"OK",
"total_fee":100.00, //order total price
"app_id":"bf18ac2de375095d63428134e44d1867",
"sub_code":"SUCCESS",
"receipt_fee":100.00, //receipt price
"signature":"be3fae4d68fec9c444fde821659bce69",
"buyer_pay_fee":100.00//price that buyer actually paid
}

- When queried failed:

{"code ":"14001","msg":"order not exists"}

Other code ID and its explanations are the same as the table showing above.

8.3.4 Acquire User Info

- Method to call
  - void GetUserAPI()
  - Explanation: acquire logged in users' information
- Callback method
  - void UserInfoCallback (string userInfo)
  - Return values explanation: userInfo——return login information from backend. It is a Json string that has not been processed.
  - Example format when queried successfully:

{"ret_code":"0000",
"data":{
"aboutme":"",
"birthday":1460476800000,
"phone":"13100000000",
"username":"Admin",
"email":"",
"gender":"male",
"lastname":"",
"openid":"4f3148bdc34d9bca104927729a173b64",
"firstname":"",
"avatar":"http://172.31.83.11/upload/6dd6ee103714e967846c3d38ae48d511",

"signature":"14a25d7219d8dfc91e55f63286ae5c0a",

"country":"China",

"city":""

},

"ret_msg":"called successfully"

}

- When queried failed:

{

"ret_code":"00003000",

"ret_msg":"signature verify failed"

}

Other ret_code ID and its explanations:

0000	request successfully
00020000	database operate failed
9999	system error
00001000	parameter error
00002000	data analyze failed
00003000	signature verify failed
00003001	time verify failed
00060000	user not found
00060001	user password wrong
00060002	user login unknown error

00061000	user token not found
00061001	user token verify failed
00061002	user token unknown error
00070001	app verify error
00071001	app secret key verify failed
00080001	OAUTH_CODE verify failed
00090001	REFRESH_TOKEN verify failed
00100001	ACCESS_TOKEN verify failed
00110001	SCOPE verify failed

8.4 Developer-Backend Interaction

After payment complete, payment system will send related payment result and user information to merchants. Merchants need to be able to receive and process the data and return response.

When backend notify to interact, it will be considered as failed if payment system received neither success nor timed out from merchant. Therewith, payment system will recreate notification through certain strategy to increase success possibility as much as it can, but cannot ensure success eventually.

Notice: Merchant could receive the same notification more than one time. Therefore, merchant system must be able to process repeat notification correctly.

The recommend way to do is: check status of corresponding transaction data to see whether that notification has been processed first. If not, process it. Otherwise,

return result as success directly. Before checking or processing transaction data, data lock should be used to do concurrency control in order to avoid data confusion caused by reentrant functions.

Special remind: merchant system must verify signature on content of notification of payment result to avoid "fake notification" caused by data breach and money loss. More details to see 8.4.3 Signature Verification.

Interface link is "notify_url" in the parameter called "payOrderJson" when calling payment interface. If this link cannot be accessed, merchant will not be able to receive notification. Notify URL has to be accessible directly without any parameters.

Backend of merchant need implement following interface to receive requests from Pico backend to acquire payment results and user information from Pico payment system.

Name	Callback interface for payment results
Request type	POST
Request URL	Payment, PayOrder input parameter: notify_url
Request format	JSON
Return format	JSON
If need login	Yes
Request parameters	More details to see 8.4.1 notification

	parameters of payment results

Examples of request parameters

Return parameters

Parameter	Type	Description
ret_code	string	Wrong code
ret_msg	string	Wrong information string

See more at 8.4.2 return results

Examples of return parameters

```
{
"ret_code": “SUCCESS”,
"ret_msg": "OK"
}
```

8.4.1Notification Parameters

Field Name	Variable Name	Required	Type	Description
Return status code	ret_code	Yes	String	SUCCESS/FAIL Communication identification, not transaction identification. Need to check result_code to see transaction result

Return information	ret_msg	No	String	Return information, it is error reason if not empty .Signature verify failed or wrong format of parameters
Wrong code	sub_code	No	String	Error code
Wrong code description	sub_msg	No	String	Description for returned error
Pico payment order	trade_no	Yes	String	Pico payment order number
Merchant payment order	out_trade_no	Yes	String	Order number inside merchant system
APP ID	app_id	Yes	String	APP ID approved by platform
Merchant ID	mch_id	Yes	String	Merchant ID distributed by payment
User ID	open_id	Yes	String	The only identification for user under app id
Device ID	device_id	No	String	Terminal device number
Random String	nonce_str	Yes	String	Random string, no long than 32 bit. Recommend to use random number generation algorithm.
Trade type	trade_type	Yes	String	Payment type

Currency type	fee_type	Yes	String	Currency type

Field Name	Variable Name	Required	Type	Description
Total amount	total_fee	Yes	String	Total amount
Receive amount	receipt_fee	Yes	String	Receive amount
Buyer paid amount	buyer_pay_fee	No	String	Buyer actually paid amount
Coupon or other discount amount	coupon_fee	No	String	Coupon or other discount amount
Merchant data package	attach	No	String	Merchant data package, returned as the same
Payment complete time	pay_time	Yes	String	Payment complete time, format is yyyy-MM-dd HH:mm:ss

8.4.2 Return Results

Field Name	Variable Name	Required	Type	Description
Returned status code	ret_code	Yes	String	SUCCESS/FAIL SUCCESS means merchant

				receive and verify notification successfully
Returned information	ret_msg	No	String	Return information, it is error reason if not empty Signature verify failed or wrong format of parameters

Failed result

i.e.:

{"ret_code":" SUCCESS","ret_msg":"OK"}

8.4.3 Signature Verification

The rules for signature verification are:

1. For the table of returned parameters, remove signature parameter first, and add key=" app_secret" , value=paykey. And then sort them based on key value, and make sure separate parameters with &. At last, encode with MD5.

2. Compare string after encryption with acquired signature.

Signature method as following:

```
/**
* result: map collection of acquired data
* paykey: is the paykey on developer platform
*/
publi cstatic String createSign(Map<String, Object> result, String paykey)
{
    if(result==null||result.size()==0)
    return null;
    result.put("app_secret",paykey); //1. Add key = “app_secret”,value=payke
    String sign = result.get("signature");//2. Save value of signature to verify
    result.remove("signature"); //3. Remove signature parameter
```

```
        String[] tmp =new String[result.size()];
        int i =0;
        for(String key :result.keySet())
        {
            tmp[i++]= key;
        }
        Arrays.sort(tmp); //4. Sort it
        String sign ="";
        for(String string : tmp)
        {
            if(m.get(string)==null)
            continue;
            sign += string +"="+ URLEncoder.encode(m.get(string).toString(),"utf-8")+"&";
        }
        if(sign.endsWith("&"))
            sign = sign.substring(0, sign.length()-1);
        Log.i(TAG,"createSign: "+sign);
        String localSign = MD5.MD5(sign); //5. Generate string after MD5 encryption
        return localSign.equal(sign);//6. Verify with signature in 2
}
```

9 Pvr_UnitySDK API

9.1 Pvr_UnitySDK Prefab and Script function description

Pvr_UnitySDK Prefab

- To control the position of the camera in a Unity scene: Pvr_UnitySDK.prefab
- The prefab organizes the related layers and function of Head

Figure 9.1 Pvr_UnitySDK.prefab hierarchy

- is suggested to use Pvr_UnitySDK prefab in Unity Editor as follows:

- Create a new scene
- Delete the Main Camera in the new created scene
- Use Pvr_UnitySDK prefab under the directory of \Assets\ PicoVRSDK \Prefabs

Scripts directory structure

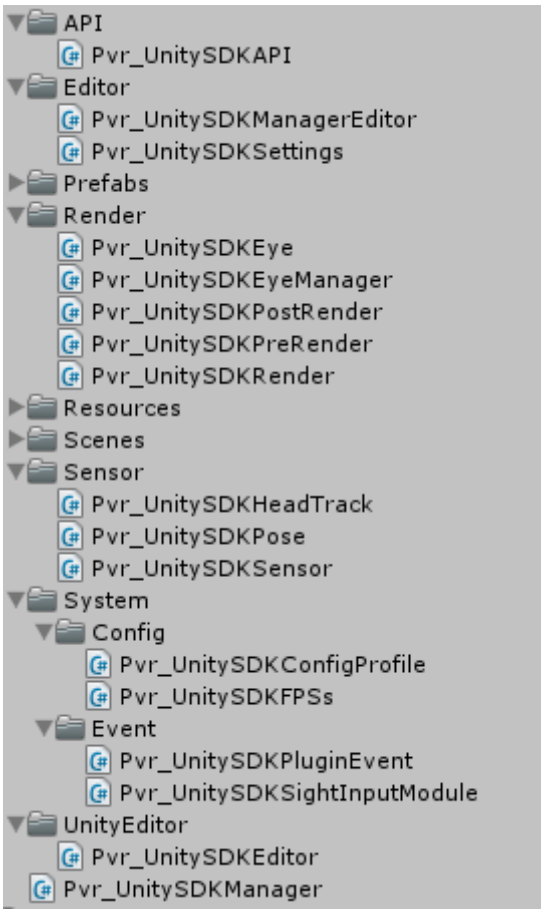


Figure 9.2 Scripts Directory Structure

9.1.1 Pvr_UnitySDKConfigProfile Script

- Directory : \Assets\Pvr_UnitySDK\System\Config\
- Main Function : The file is used for parameters configuration

9.1.2 Pvr_UnitySDKFPSs Script

- Directory : \Assets\ Pvr_UnitySDK\System \Config\
- Main Function：To show the fps in the scene

9.1.3 Pvr_UnitySDKSensor Script

- Directory : \Assets\Pvr_UnitySDK\Sensor\
- Main Function：Sensor function implementation

9.1.4 Pvr_UnitySDKPluginEvent Script

- Directory : \Assets\Pvr_UnitySDK\Event\
- Main Function：the interaction with bottom layer, including rendering, process control,etc

9.1.5 Pvr_UnitySDKSightInputModule Script

- Directory : \Assets\Pvr_UnitySDK\Event\
- Main Function：Cursor pickup
- Editable Parameters：

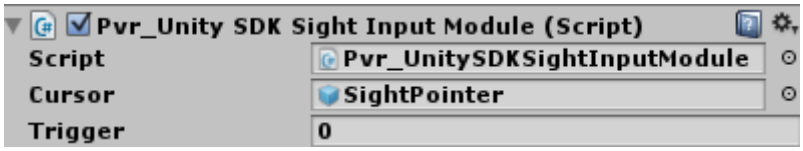


Figure 9.3 SightInputModule Editable Parameters

- Cursor： Cursor Object

9.1.6 Pvr_UnitySDKEye Script

- Directory : \Assets\Pvr_UnitySDK\Render\
- Main Function：Control and manage the left and right cameras
- Editable Parameters：

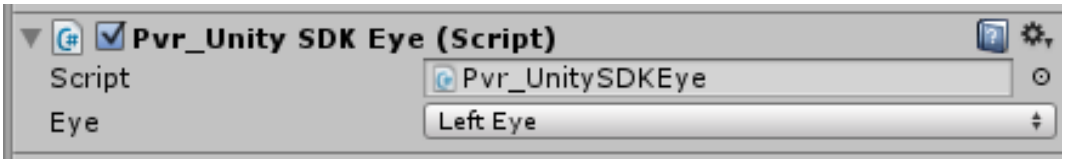


Figure 9.4 PicoVR Eye Editable Parameters

- Eye： Mark as left or right eye

9.1.7 Pvr_UnitySDKEyeManager Script

- Directory : \Assets\Pvr_UnitySDK\Render\
- Main Function：Manage the scene stereo for both eyes

9.1.8 Pvr_UnitySDKHeadTrack Script

- Directory : Directory : \Assets\Pvr_UnitySDK\Sensor\
- Main Function：Track the head rotation and head position
- Editable Parameters：

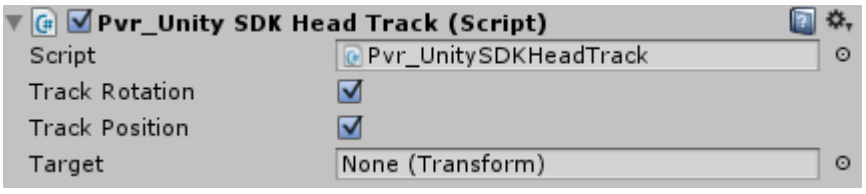


Figure 9.5 PicoVR HeadTrack Editable Parameters

- Track Rotation： Switch of Track Rotation

- Track Position： Switch of Track Position
- Target： 3D coordinate info
- Update Early： The track moment of each frame

9.1.9 Pvr_UnitySDKPose Script

- Directory : Directory : \Assets\Pvr_UnitySDK\Sensor\
- Main Function : Define and track head rotation and head position , ordinary class.

9.1.10 Pvr_UnitySDKManager Script

- Directory : \Assets\Pvr_UnitySDK\
- Main Function：main controller of SDK camera
- Editable Parameters：

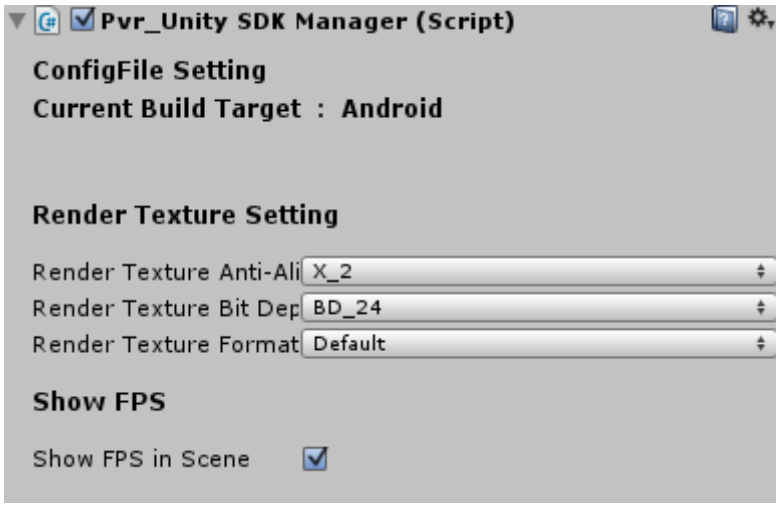


Figure 9.6 Rendertexture Setting

- Render Texture Anti-Aliasing： Render Texture anti-aliasing setting
- Render Texture Bit Depth： Render Texture bit depth setting

- Render Texture Format： Render Texture format setting
- Render Texture Size： Render Texture size

Render Texture Size modify the default does not take effect, if you want to modify Please open the Pvr_UnitySDKRender.cs script set the following statement:

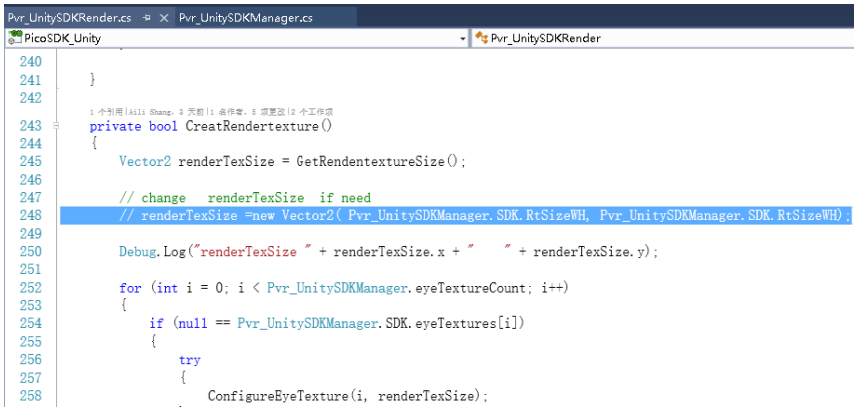


Figure 9.7 Rendertexture size settings

- Show FPS: Show FPS tool
- Screen Fade: Screen fade effect tool

9.2 Unity API

9.2.1 UPvr_GetUnitySDKVersion

- ✧ public static string UPvr_GetUnitySDKVersion ()
- ✧ Description：Get the SDK version
- ✧ Input Parameters：Null
- ✧ Return Value：string
- ✧ Call：Pvr_UnitySDKAPI.System. UPvr_GetSDKVersion ()

9.2.2 Sensor-Related API

- public static int UPvr_StartSensor(int index)
- Description：Turn on sensor tracking
- Input Parameters：index （0：Main sensor ， 1 the first vice sensor）
- Return Value：int 0 suceed ; 1 fail
- Call：Pvr_UnitySDKAPI. Sensor. UPvr_StartSensor(index)
- Note: The main sensor is DK/DKS with HMD head sensor and Goblin head sensor . The first vice sensor is DK/DKS handle sensor

- public static int UPvr_ResetSensor(int index)
- Description：To reset sensor tracking
- Input Parameters：index （0：Main sensor ， 1 the first vice sensor）
- Return Value：int 0 suceed; 1 fail
- Call: Pvr_UnitySDKAPI. Sensor. UPvr_ResetSensor(index)
- Additional instructions：In this method, when Enable6Dof is True, What is called is Pvr_OptionalResetSensor(int index, Pvr_UnitySDKManager.SDK.resetRot, Pvr_UnitySDKManager.SDK.resetPos, resetRot), the default value of resetPos is 0, Position and Rotation are not Reset, but according to their own needs to modify the use of. The modification is as follows,In Pvr_UnitySDKManager.cs's Awake () method, add resetRot = 1; resetPos = 1;

- public static int UPvr_StopSensor(int index)
- Description：To stop sensor tracking
- Input Parameters：index （0：main sensor ， 1 the first sensor）
- Return Value：int 0 suceed; 1 fail
- Call : Pvr_UnitySDKAPI. Sensor. UPvr_StopSensor(index)

9.2.3 Pico Controller Related

- public static void StartScan ()
- Description：To scan Pico Controller
- Input Parameters：null
- Return Value：null
- Call: Pvr_ControllerManager.StartScan（ ）

- public static void StopScan()
- Description：To stop scanning Pico Controller
- Input Parameters：null
- Return Value：null
- Call: Pvr_ControllerManager.StopScan()

- public static void ConnectBLE ()
- Description：To connect to Pico Controller
- Input Parameters：null

- Return Value：null
- Call: Pvr_ControllerManager. ConnectBLE ()

- public static int UPvr_GetControllerPower ()
- Description：To get Pico Controller power status
- Input Parameters：null
- Return Value：1-4
- Call：Pvr_ControllerAPI. Controller .UPvr_GetControllerPower()

- public static void setHbControllerMac (string mac)
- Description：to set Pico Controller mac address
- Input Parameters：string mac
- Return Value：null
- Call: callback function，the parameter(string mac) can be used directly

- public static void setHbServiceBindState (string state)
- Description：to get Pico Controller Service binding state
- Input Parameters：string state
- Return Value：null
- Call: callback function，the parameter (string state) can be used directly,

  0-unbinding，1-binding，2-unknown

- public static bool isHbServiceExisted ()

- Description：to check if Pico Controller Service exists
- Input Parameters：null
- Return Value：true:exist，false: not exist

- public static ControllerState UPvr_GetControllerState ()
- Description：to get Pico Controller connection state
- Input Parameters：null
- Return Value：connection state

- public static void ResetController()
- Description：Reset the HummingBird handle gesture
- Input Parameters：null
- Return Value：null

- public static void RebackToLauncher ()
- Description：Return to launcher screen
- Input Parameters：null
- Return Value：null
- Call: call in Pvr_ControllerManager class

- public static bool StartUpgrade ()
- Description：to start upgrade
- Input Parameters：null
- Return Value：bool, succeed or failed

- Call: call in Pvr_ControllerManager class

- public static string GetBLEImageType ()
- Description：to get BLE type
- Input Parameters：null
- Return Value：BLE type
- Call: call in Pvr_ControllerManager class

- public static long GetBLEVersion ()
- Description：to get BLE version
- Input Parameters：null
- Return Value：BLE version
- Call: call in Pvr_ControllerManager class

- public static long GetFileVersion ()
- Description：to get File version
- Input Parameters：null
- Return Value：File version
- Call: call in Pvr_ControllerManager class

- public static void setupdateFailed ()
- Description：callback of update failed
- Input Parameters：null
- Return Value：null

- Call: call in Pvr_ControllerManager class

- public static void setupdateSuccess ()
- Description：callback of update success
- Input Parameters：null
- Return Value：null
- Call: call in Pvr_ControllerManager class

- public static void setupdateProgress (string progress)
- Description：callback of update progress
- Input Parameters：string progress
- Return Value：null
- Call: call in Pvr_ControllerManager class ，string progress value from 0 to 100

- public static Quaternion UPvr_GetControllerQUA ()
- Description：to get the rotation quaternion of Pico Controller
- Input Parameters：null
- Return Value：Pico Controller rotation quaternion
- Call: call in Pvr_ControllerAPI

- Public static bool UPvr_GetKey (Pvr_KeyCode key)
- Description：to get key state
- Input Parameters： Pvr_KeyCode

- Return Value：bool, success or failed
- Call: call in Pvr_ControllerAPI

- Public static bool UPvr_GetKeyDown (Pvr_KeyCode key)
- Description：to get key state
- Input Parameters： Pvr_KeyCode
- Return Value：bool, success or failed
- Call: call in Pvr_ControllerAPI

- Public static bool UPvr_GetKeyUp (Pvr_KeyCode key)
- Description：to get key state
- Input Parameters： Pvr_KeyCode
- Return Value：bool, success or failed
- Call: call in Pvr_ControllerAPI

- Public static bool UPvr_GetKeyLongPressed (Pvr_KeyCode key)
- Description：to get key state
- Input Parameters： Pvr_KeyCode
- Return Value：bool, success or failed
- Call: call in Pvr_ControllerAPI

- Public static bool UPvrr_IsTouching ()
- Description：to get is or not touching the touchpad
- Input Parameters： null

- Return Value：bool, success or failed
- Call: call in Pvr_ControllerAPI

- Public static bool UPvr_GetSlipDirection (Pvr_SlipDirection dir)
- Description：to get slide state
- Input Parameters： Pvr_SlipDirection
- Return Value：bool, success or failed
- Call: call in Pvr_ControllerAPI

- Public static bool int UPvr_GetTouchPadPosition (int tp)
- Description：to get touchpad position
- Input Parameters： 0 or 1
- Return Value：0 will return X , 1 will return Y
- Call: call in Pvr_ControllerAPI

- Public static void AutoConnectHbController (int scans)
- Description：Automatic search connection handle
- Input Parameters： Search time in units per millisecond
- Return Value：null
- Call: call in Pvr_ControllerManager

- Public static void setHbAutoConnectState (string state)

- Description：Callback of automatic search handle
- Input Parameters： null
- Return Value：State

```
//UNKNOW = -1; // Default value
//NO_DEVICE = 0;// No scan to the controller
//ONLY_ONE = 1;// Scan only to one controller
//MORE_THAN_ONE = 2;// Scan to more than one controller
//LAST_CONNECTED = 3;// Scan to the last connected controller
//FACTORY_DEFAULT = 4;// Scan to factory bound controller
```

(temporarily unused)

- Call: call in Pvr_ ControllerManager

9.2.4 Battery,Volume,Brightness Related API

- public bool UPvr_InitBatteryVolClass()
- Description：to initialize the battery, volume, brightness service
- Input Parameters：null
- Return Value：bool , success or failed
- Call：please call this API before accessing battery, volume, brightness related API

- public bool UPvr_StartAudioReceiver ()

- Description：to start audio receiver
- Input Parameters：null
- Return Value：bool, success or failed

- public void UPvr_SetAudio(string s)
- Description：notification to Unity when system volume changes
- Input Parameters：volume value after change
- Return Value：null

  Tip：This API is called by Android bottom layer, no need to call in Unity, If some operations are expected when volume change, write the code into this method

- public bool UPvr_StopAudioReceiver ()
- Description：to stop audio receiver
- Input Parameters：null
- Return Value：bool, success or failed

- public bool UPvr_StartBatteryReceiver ()
- Description：to start battery receiver
- Input Parameters：null
- Return Value：bool, success or failed

- public void UPvr_SetBattery(string s)
- Description：notification to Unity when system battery changes
- Input Parameters：volume value after change（0.00~1.00）
- Return Value：null
- Tip：This API is called by Android bottom layer, no need to call in Unity, If some operations are expected when battery change, write the code into this method

- public bool UPvr_StopBatteryReceiver ()
- Description：to stop battery receiver
- Input Parameters：null
- Return Value：bool, success or failed

- public int UPvr_GetMaxVolumeNumber ()
- Description：to get max volume number
- Input Parameters：null
- Return Value：int, max volume number

- public int UPvr_GetCurrentVolumeNumber ()
- Description：to get current volume number
- Input Parameters：null
- Return Value：int, current volume number(0~15)

- public bool UPvr_VolumeUp ()
- Description：to increase volume
- Input Parameters：null
- Return Value：bool, success or failed

- public bool UPvr_VolumeDown ()
- Description：to decrease volume
- Input Parameters：null
- Return Value：bool, success or failed

- public bool UPvr_SetVolumeNum(int volume)
- Description：to set volume
- Input Parameters：int, expected volume number(0~15)
- Return Value：bool, success or failed

- public bool UPvr_SetBrightness(int brightness)
- Description：to set brightness
- Input Parameters：int, expected brightness number(0~255)
- Return Value：bool, success or failed

  Tip：For Android phone only, not for Pico Neo

- public int UPvr_GetCurrentBrightness()
- Description：to get current brightness
- Input Parameters：null
- Return Value：int, current brightness（0~255）

  Tip：For Android phone only, not for Pico Neo

- public bool UPvr_IsHmdExist()
- Description：to check if the device is HMD
- Input Parameters：null
- Return Value：true HMD ; false not HMD

- public int UPvr_GetHmdScreenBrightness()
- Description：to get HMD screen brightness
- Input Parameters：null
- Return Value：int, screen brightness（0~255）

  Tip：For Pico Neo only, not for Android phone

- public bool UPvr_SetHmdScreenBrightness(int brightness)
- Description：to set HMD screen brightness
- Input Parameters：int, screen brightness（0~255）
- Return Value：bool, success or failed

  Tip：For Pico Neo only, not for Android phone

- public bool UPvr_SetCommonBrightness(int brightness)
- Description：to set brightness
- Input Parameters：int, expected brightness number(0~255)
- Return Value：bool, success or failed

  Tip：For Android phone and Pico Neo

- public int UPvr_GetCommonBrightness ()
- Description：to get current brightness
- Input Parameters：null
- Return Value：int, current brightness（0~255 ）

  Tip：For Android phone and Pico Neo

9.2.5 Surrounding Sound Effect (For Pico Neo)

- public void UPvr_OpenEffects()
- Description：to switch on sound effect
- Input Parameters：null
- Return Value：null

- public void UPvr_CloseEffects()
- Description：to switch off sound effect
- Input Parameters：null

- Return Value：null

- public void UPvr_EnableSurround()
- Description：to enable surrounding sound effect
- Input Parameters：null
- Return Value：null

- public void UPvr_SetSurroundroomType(int type)
- Description：to simulate the room size of surrounding sound
- Input Parameters：int, type:1 small、 2 medium 、3 large（the API should be called after the surrounding effect enabled）
- Return Value：null

- public void UPvr_OpenRoomcharacteristics()
- Description： to open simulation of the damped character of room（the API should be called after the surrounding effect enabled）
- Input Parameters：null
- Return Value：null

- public void UPvr_CloseRoomcharacteristics()
- Description：to close simulation of the damped character of room
- Input Parameters：null

- Return Value：null

- public void UPvr_EnableReverb()
- Description：to enable reverb effect
- Input Parameters：null
- Return Value：null

9.2.6 Audio Play Related (For Pico Neo)

- public void UPvr_StartAudioEffect(String audioFile,bool isSdcard)
- Description：to set the absolute path of audio file
- Input Parameters :audioFile ,the absolute Android system path of audio file.

  For example: string audioFile = "/sdcard/ test.mp3"; string audioFile = "/system/newdir/GuitarLoop.wav"; , isSdcard is reserved parameter, which can be set true.
- Return Value：null

  Tip：For Pico Neo, the path is required before play

- public void UPvr_StopAudioEffect()
- Description：to stop audio effect
- Input Parameters：null
- Return Value：null

- public void UPvr_ReleaseAudioEffect()
- Description :To release audio effect, no audio file can be loaded after called, it is recommended to call before exit the app.
- Input Parameters：null
- Return Value：null

9.2.7 Psensor State（For Pico Neo）

- public static int UPvr_GetPsensorState()
- Description：to get state of Psensor
- Input Parameters：null
- Return Value：Psensor blocked 0 ，unblocked 1
- Call: Pvr_UnitySDKAPI. Sensor. UPvr_GetPsensorState()

9.2.8 Controller Vibrate Related（For Pico Neo）

- public static bool UPvr_HasControllerVibrator()
- Description：Whether has controller vibrator function
- Input Parameters：null
- Return Value：bool
- Pvr_UnitySDKAPI. Haptics. UPvr_HasControllerVibrator ()

- public static void UPvr_SetControllerVibrateMode(int[] pattern, int length, int repeat)

- ✧ Description：Set Controller Vibrate Mode
- ✧ Input Parameters：

  int[] pattern =new int[5] { vibrationDuration,silienceDuration

  ,repeat_times, vibrationStrength, whichHaptic }

  vibrationDuration：vibrate Duration（milliseconds）

  silienceDuration：silience Duration（milliseconds）

  repeat_times：repeat times

  vibrationStrength：vibration Strength（ 0—127 range）

  whichHaptic： 1，left 2，right 3，both left and right

  length：patterns length，must input 5

  repeat：input 1
- ✧ Return Value：null
- ✧ Pvr_UnitySDKAPI. Haptics. UPvr_SetControllerVibrateMode(pattern, length, repeat)

- ✧ public static void UPvr_SetControllerVibrateTime(int milliseconds)
- ✧ Description：Set Controller Vibrate Time
- ✧ Input Parameters：Controller Vibrate Time
- ✧ Return Value：null
- ✧ Pvr_UnitySDKAPI. Haptics. UPvr_SetControllerVibrateTime (milliseconds)

- ✧ public static void UPvr_CancelControllerVibrate()

- Description：Cancel Controller Vibrate
- Input Parameters：null
- Return Value：null
- Pvr_UnitySDKAPI. Haptics. UPvr_CancelControllerVibrate ()

9.2.9 Get DeviceMode Infomation

- public static DeviceMode UPvr_GetDeviceMode ()
- Description：Get DeviceMode Infomation
- Input Parameters：null
- Return Value：enum DeviceMode，include FalconDKS，FalconDK, Goblin,Other Psensor
- Call: Pvr_UnitySDKAPI. System. UPvr_GetDeviceMode ()

- public static string UPvr_GetDeviceSN ()
- Description：Get Device Serial Number
- Input Parameters：null
- Return Value：Serial Number
- Pvr_UnitySDKAPI. System. UPvr_ GetDeviceSN ()

- public static void UPvr_ShutDown()
- Description：Shut down device
- Input Parameters：null

- ✧ Return Value：null
- ✧ Pvr_UnitySDKAPI. System. UPvr_ ShutDown ()

- ✧ public static void UPvr_ Reboot ()
- ✧ Description：Reboot device
- ✧ Input Parameters：null
- ✧ Return Value：null
- ✧ Pvr_UnitySDKAPI. System. UPvr_ Reboot ()

- ✧ public static void UPvr_ Sleep ()
- ✧ Description：Sleep device
- ✧ Input Parameters：null
- ✧ Return Value：null
- ✧ Pvr_UnitySDKAPI. System. UPvr_ Sleep ()

Tip：With UPvr_ShutDown, UPvr_Reboot, and UPvr_Sleep interfaces, you need to add the following permissions in AndroidManifest and require system signatures:

android:sharedUserId="android.uid.system"

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

FAQ

Q: How to set Camera Culling Mask?

A: The "Culling Mask" configurations for LeftEye/RightEye of PicoVR can be

set in Inspector:

1. If your application has adapted to Cardboard SDK and used "Toggle Culling Mask", this can be done by changing the configurations of Camera directly.

2. If the "Toggle Culling Mask" need to be changed in code level, please refer to the configurations of normal camera.

3. When it is necessary to see different scenes for different eyes, the displaying layers for different eyes could be controlled by Culling Mask of PicoVR

Q; How to exit in PicoNeo ?

A: Developers can implement their own functions or find the QuitGame.cs script in Assets\PicoVRSDK\Scripts ,and drag it to the PicoVR's GameObject. Then press the Back key to exit in run time. In this way, developers need to release loaded resources in games or applications by themselves.

Q: How are the keys on Pico Neo controller /Goblin mapped to the keys in Unity?

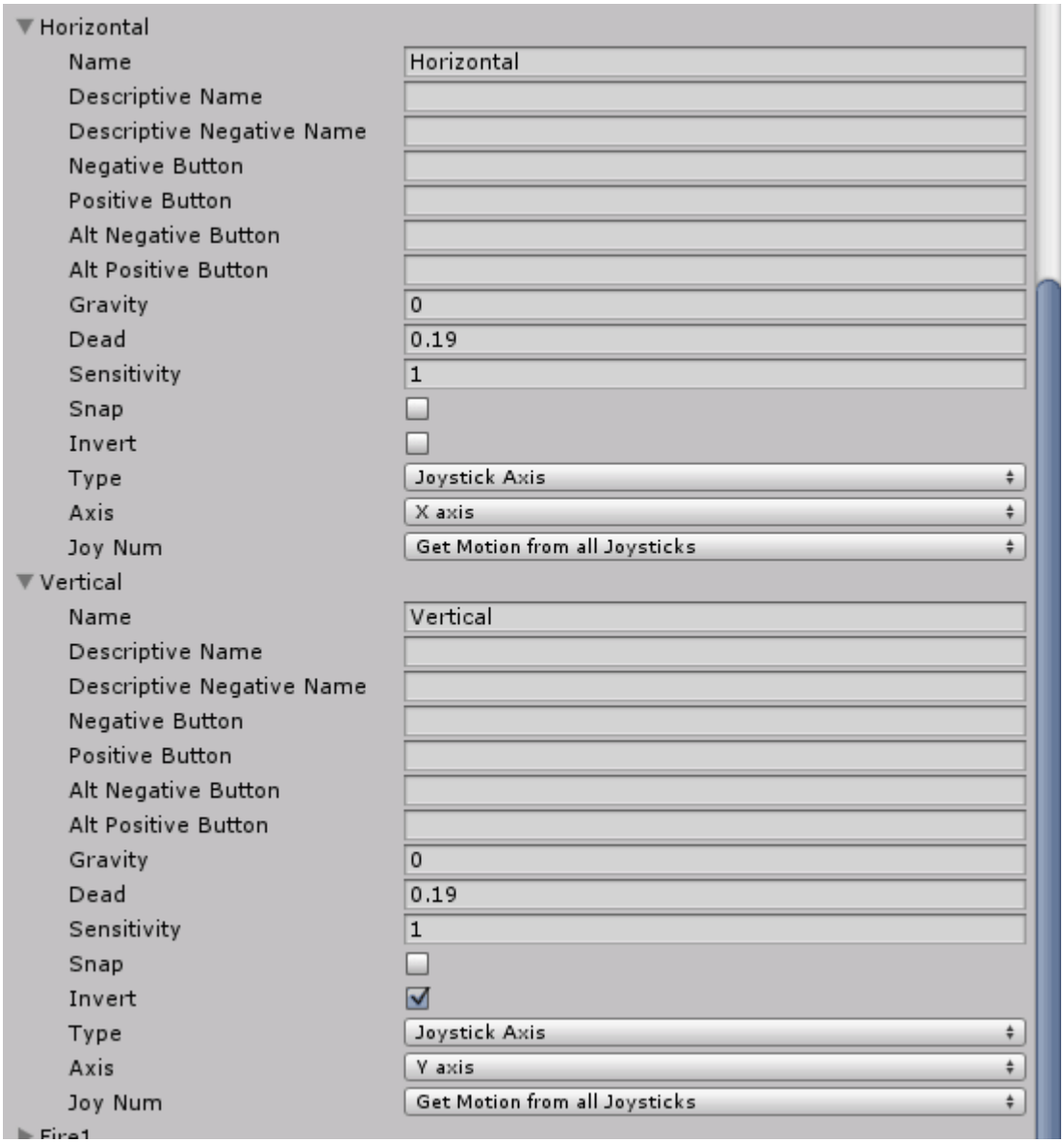
A: You can refer to the following mapping table

The mapping table of keys between Pico Neo Controller and Unity

Note: Goblin's enter key corresponds to JoystickButton0

The input settings in Unity are shown below

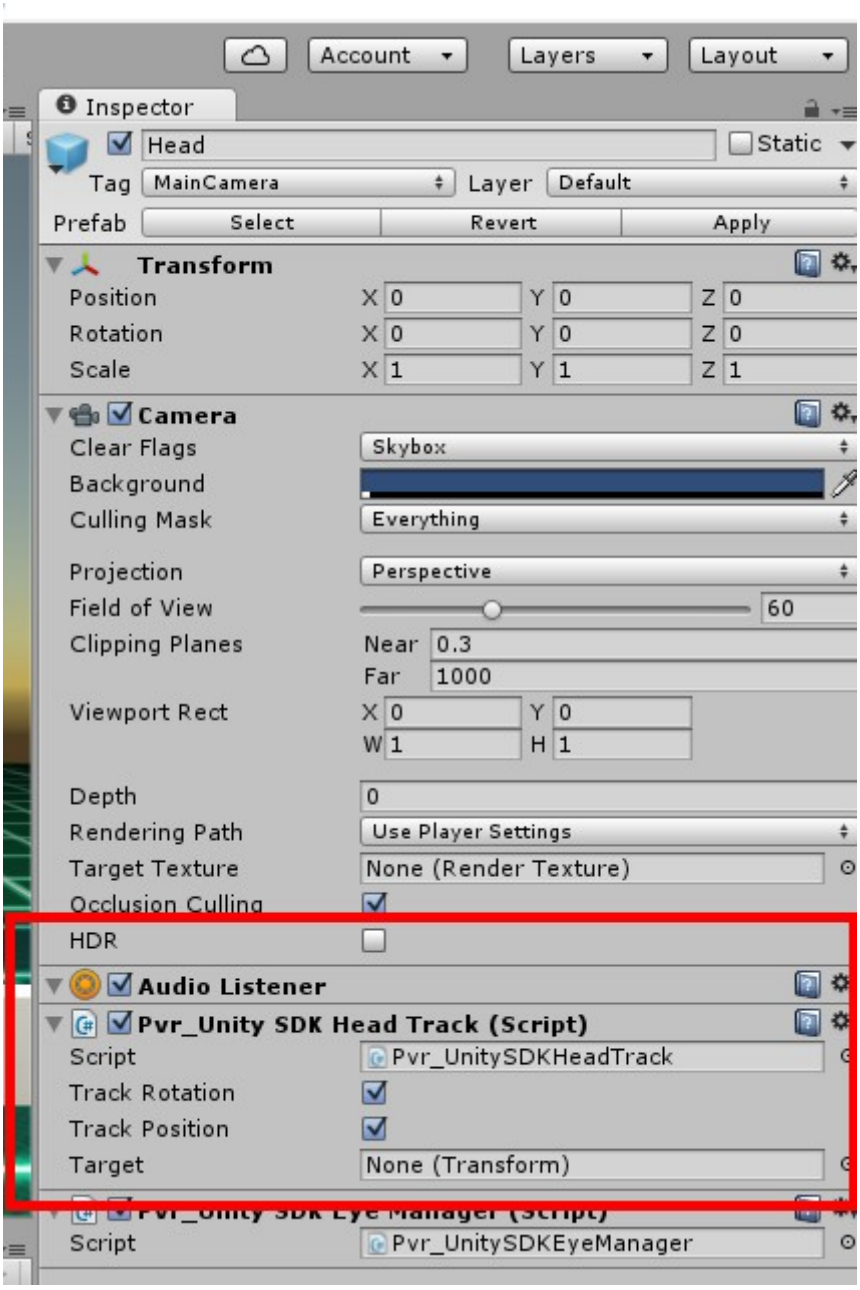
number	Key	Unity corresponding value 1	Unity corresponding value 2
1	A	JoystickButton0	Joystick1Button0
2	B	JoystickButton1	Joystick1Button1
3	X	JoystickButton2	Joystick1Button2
4	Y	JoystickButton3	Joystick1Button3
5	Menu	Menu	/
6	Return	Escape	/
7	Left Trigger	JoystickButton4	Joystick1Button4
8	Right Trigger	JoystickButton5	Joystick1Button5
9	power	Null	Null
10	Horizontal Axis	Input.GetAxis("Mouse X")	Input.GetAxis("Horizontal")
11	Vertical Axis	Input.GetAxis("Mouse Y")	Input.GetAxis("Vertical")
12	Pico	Null	Null



Q: How to use the sensor to control other game object in scene?

A: Follow these steps:

1. Uncheck PicoVRHeadTrack script on PicoVR/head
2. Add PicoVRHeadTrack script on other game object



Q: What are the requriements for the frame rate and scene if we want to get smooth effect?

A: **Frame Rate:**

Application frame rate of not less than 30 frames, it is recommended more than 45 frames.

Scene :

Model Tris should be within 100,000

Model Verts should be within 100,000

Unity light source should be within 50

Unity Particle system should be within 50

Unity shader should be as effective as possible.

Optimization example：

Reduce the Tris amount under the premise of graphic quality

Use share material and animation

Use compressed texture format

Use as less light effect as possible

Use Static Batching Utility

Script code optimization, reduce unnecessary resource consumption

Q：How to replace the application opens loading animation (For Goblin)

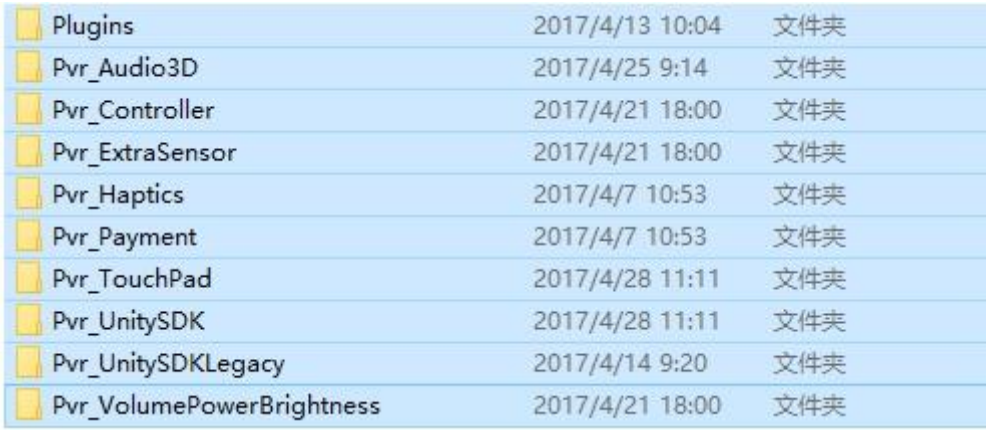
A：Open the AndroidManifest put platform_logo TAB value to 1。

```
<meta-data android:name="platform_logo" android:value="1" />
<meta-data android:name="platform_high" android:value="0" />
```

Q: How to replant the new SDK into the project built with the old version?

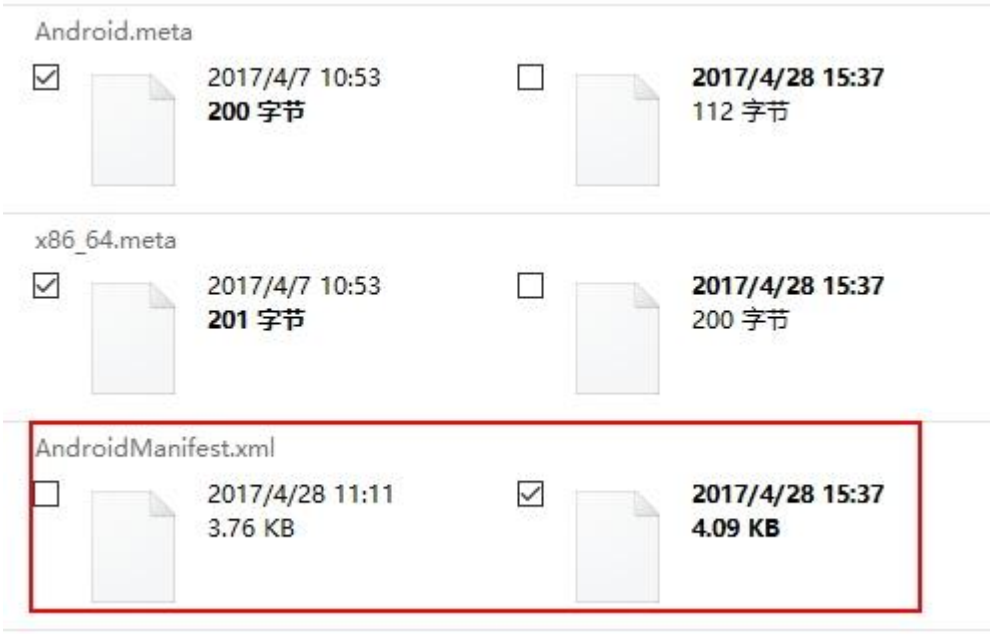
A：It is recommended to use version 2.x, the new SDK is compatible with old ones, the method as below:

1．Copy folders under /PicoSDK_Unity/Assets/

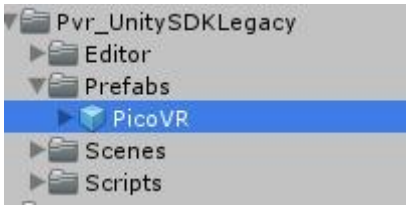


2.Paste to /Asstes directory of original project, select Let me decide.

3. As the figure below, keep the original AndroidManifest.xml, and replace all other folders and files

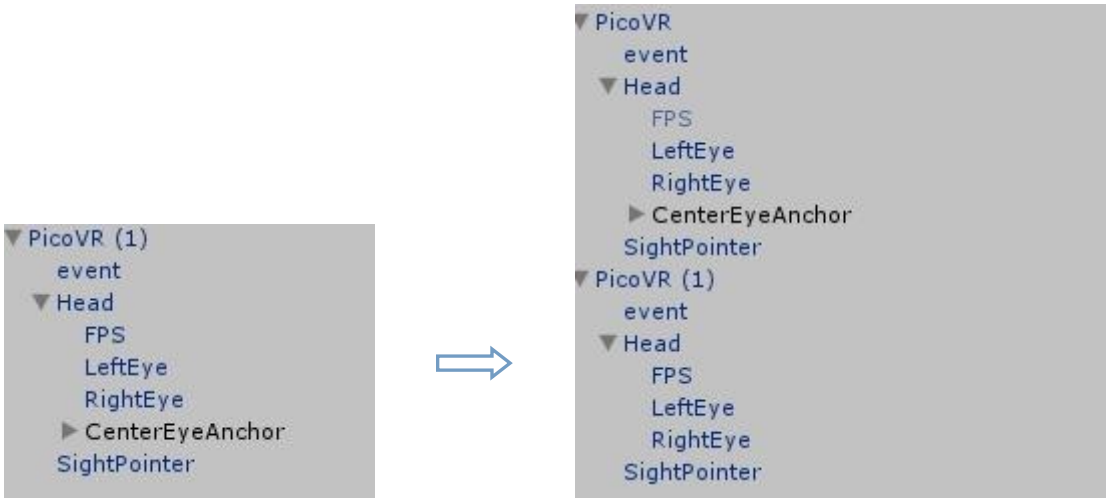


4.Put PicoVR prefab from /Pvr_UnitySDKLegacy/Prefabs into the Scene

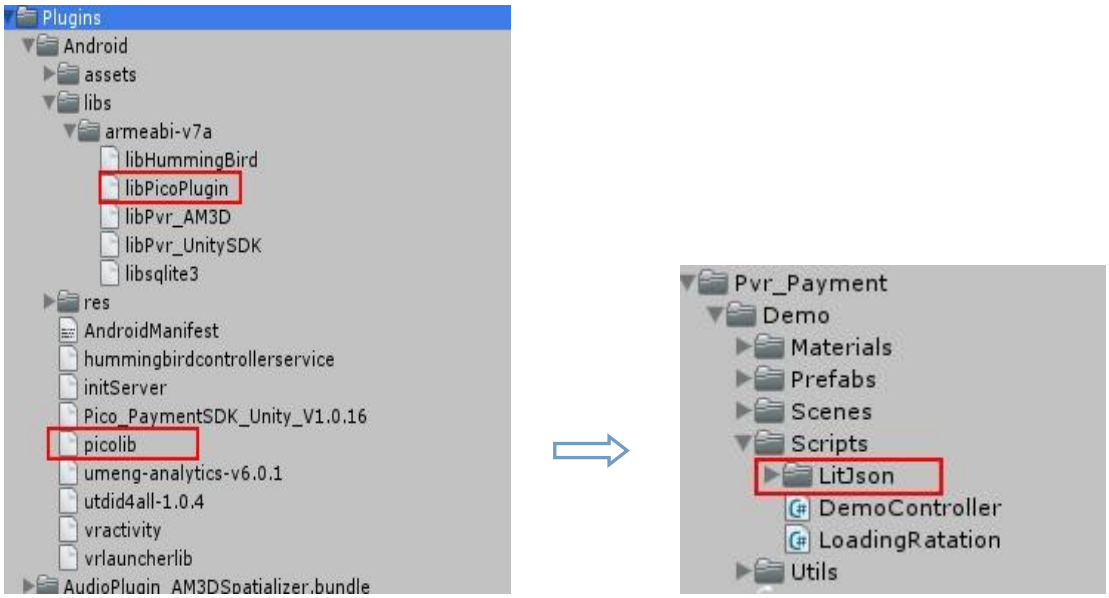


5. Put the CenterEyeAnchor from PicoVR into the new hierarchy, and

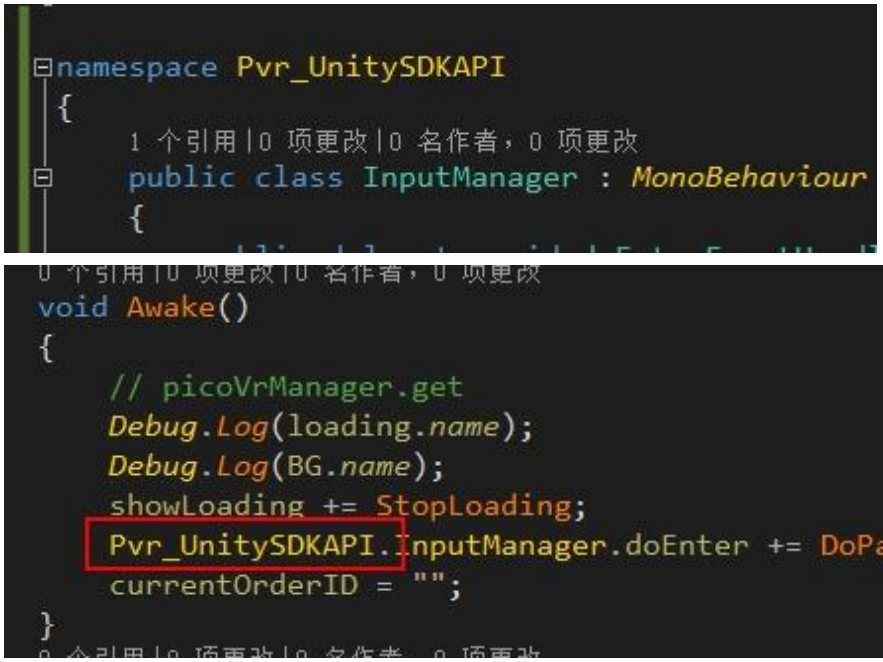
delete the old PicoVR camera



6.Delete PicoVRSDK folder, delete libPicoPlugin.so and Picolib.jar in Plugins, delete LitJson folder in Pvr_Payment(if there is no LitJson in old project, ignore this step)



7. A new namespace is required in /vr_Payment/Demo/Utils/Input-Manager.cs since the InputManager from Payment interrupts with the one from PUI. Add Pvr_UnitySDKAPI in Pvr_Payment/Demo/Scripts/Demo-Controller.cs as the figure below.



```
namespace Pvr_UnitySDKAPI
{
    1 个引用|0 项更改|0 名作者，0 项更改
    public class InputManager : MonoBehaviour
    {
        0 个引用|0 项更改|0 名作者，0 项更改
        void Awake()
        {
            // picoVrManager.get
            Debug.Log(loading.name);
            Debug.Log(BG.name);
            showLoading += StopLoading;
            Pvr_UnitySDKAPI.InputManager.doEnter += DoPayByCode;
            currentOrderID = "";
        }
```

8.Delete the last two lines in AndroidManifest.xml

```
<service android:name="com.picovr.picovrlib.ble.BluetoothLeService" />
<service android:name="com.picovr.picovrlib.service.LarkConnectService" />
```

9. Now the compile should be OK

10. Addition service

Bluetooth service: add the following code in AndroidManifest.xml

```
<service android:name="com.psmart.link.spp.LarkConnectService" />
<service android:name="com.psmart.link.ble.BluetoothLeService" />
```

Controller service: add the following code in AndroidManifest.xml

```
<service
android:name="com.picovr.picovrlib.hummingbird.HummingBirdControllerService"
android:process=":remote" >
<intent-filter>
    <action android:name="com.picovr.picovrlib.humingbird.HumingBirdAIDLService"/>
</intent-filter>
```

Payment service: add the following code in AndroidManifest.xml

```
</service>
<meta-data android:name="pico_merchant_id" android:value="81"/>
<meta-data android:name="pico_app_id" android:value="5a189befeb3b33f7df101fbecffe4f98"/>
<meta-data android:name="pico_app_key" android:value="25ba00fb73343ff1ec32e1c152fff291"/>
<meta-data android:name="pico_pay_key" android:value="d828c5d3a1cc11e6b7fe008cfaf3d930"/>
<meta-data android:name="pico_redirectUrl" android:value="http://www.picovr.com"/>
```