HuaweiVR SDK2.0 Unity API说明文档

文档版本：v2.0.0

商标声明

 和其他华为商标均为华为技术有限公司的商标。

本文档提及的其他所有商标或注册商标，由各自的所有人拥有。

注意

您购买的产品、服务或特性等应受华为公司商业合同和条款的约束，本文档中描述的全部或部分产品、服务或特性可能不在您的购买或使用范围之内。除非合同另有约定，华为公司对本文档内容不做任何明示或默示的声明或保证。

由于产品版本升级或其他原因，本文档内容会不定期进行更新。除非另有约定，本文档仅作为使用指导，本文档中的所有陈述、信息和建议不构成任何明示或暗示的担保。

修改记录

日期	版本	修改描述
2017-09-08	v2.0.0	第二版

1 变量及接口函数使用说明

变量及接口均由C#代码编译生成的dll库提供，在使用这些变量及接口时，首先需要引入HVRCORE命名空间，即增加语句：using HVRCORE。HVRCORE命名空间提供HuaweiVR SDK2.0 for Unity（以下简称SDK2.0）的公共枚举、公共结构体、公共接口类等。

2 公共枚举

公共枚举	枚举说明

enum TrackStatus {
Tracking = 0,
Untracking = 1
}
	头盔跟踪状态

enum HelmetModel {
HVR_HELMET_FIRST_GEN,
HVR_HELMET_SECOND_GEN,
HVR_HELMET_NOT_FOUND,
HVR_HELMET_UNKNOWN
}
	头盔类型

enum ControllerType {
Controller3DOF,
Controller6DOF,
ControllerGaming,
ControllerSysStd,
ControllerOther
}
	控制器类型

enum ControllerEvent {
ControllerEventConnected,
ControllerEventDisconnected,
ControllerEventLowPower,
}
	控制器连接事件

enum ControllerStatus{
ControllerStatusDisconnected,
ControllerStatusScanning,
ControllerStatusConnecting,
ControllerStatusConnected,
ControllerStatusError
}
	控制器连接状态

enum ButtonType {
ButtonHome,
ButtonBack,
ButtonVolumeInc,
ButtonVolumeDec,
ButtonConfirm,
ButtonTrigger,
ButtonTouchPad
}
	控制器按钮类型

2.1 TrackStatus

TrackStatus

头盔跟踪状态，指示头部跟踪数据是否有效

枚举成员	说明
Tracking	表示头盔跟踪数据有效
Untracking	表示头盔跟踪数据无效

2.2 HelmetModel

HelmetModel

头盔类型，指示HuaweiVR头盔类型

枚举成员	说明
HVR_HELMET_FIRST_GEN	HuaweiVR1.0头盔
HVR_HELMET_SECOND_GEN	HuaweiVR2.0头盔
HVR_HELMET_NOT_FOUND	未找到头盔
HVR_HELMET_UNKNOWN	未知类型

2.3 ControllerType

ControllerType

控制器类型，指示手柄的类型（暂不支持）

枚举成员	说明
Controller3DOF	3Dof型
Controller6DOF	6Dof型
ControllerGaming	游戏型

ControllerSysStd	系统标准输入型
ControllerOther	其他类型

2.4 ControllerEvent

ControllerEvent

控制器连接事件，通过注册控制器事件句柄来监听控制器状态变化事件

枚举成员	说明
ControllerEventConnected	控制器已连接事件
ControllerEventDisconnected	控制器断开连接事件
ControllerEventLowPower	低电量告警事件

2.5 ControllerStatus

ControllerStatus

控制器连接状态，主动查询控制器当前状态

枚举成员	说明
ControllerStatusDisconnected	控制器已断开连接
ControllerStatusScanning	控制器正在扫描
ControllerStatusConnecting	控制器正在连接
ControllerStatusConnected	控制器已连接
ControllerStatusError	控制器状态错误

2.6 ButtonType

ButtonType

控制器按键类型，指示控制器包含的所有按键类型

枚举成员	说明
ButtonHome	控制器Home键（不上报）
ButtonBack	控制器返回按键
ButtonVolumeInc	控制器音量增加按键（不上报）
ButtonVolumeDec	控制器音量减小按键（不上报）
ButtonConfirm	控制器确认按键

ButtonTrigger	控制器扳机按键
ButtonTouchPad	控制器触摸板

SDK同时支持快手柄捷键和组合键：

动作	按键	按键时间	功能
单击	Home键	<1s	进入VRLauncher界面
长按	Home键	>=1s	视点重置
长按	返回键	>=3s	进入VRSetting界面
长按	Home + Trigger	>=1s	实现截图

备注

- 应用需要先正常调用手柄接口，才能正常使用手柄快捷键和组合键
- 使用截图功能时，在Unity场景需进行读写权限设置，如下：

  Player Settings… -> Other Settings -> Write Permission选择“External（SDCard）”

3 公共结构体

公共结构体	结构体说明

struct Posture{
public Quaternion rotation;
public Vector3 position;
public TrackStatus trackStatus;
}
	姿态、位置参数及跟踪状态

struct SensorData{
public Vector3 gypo;
public Vector3 accel;
public Vector3 mag;
public ulong predictTime;
}
	头盔传感器参数

struct RenderStatistics{
public double SubmitFrameRate;
public double RenderFrameRate;
public double ATWFrameCnt;
public double AvgRenderCost;
public double TimeoutFrameCnt;
public double ContinousTimeoutCnt;
}
	显示统计参数

3.1 Posture

Posture

姿态、位置参数及跟踪状态，指示头盔或手柄的旋转四元数、位置向量及跟踪状态的结构体

成员名称	成员类型	成员说明
rotation	Quaternion	旋转四元数（x,y,z,w）
position	Vector3	位置向量（x,y,z）
trackStatus	TrackStatus	头部跟踪状态

3.2 SensorData

SensorData

头盔传感器参数，指示头盔的陀螺仪数据、加速度计数据、磁力计数据、预测时间的结构体

成员名称	成员类型	成员说明
gypo	Vector3	陀螺仪参数
accel	Vector3	加速度计参数
mag	Vector3	磁力计参数
predictTime	ulong	预测时间（us）

3.3 RenderStatistics

RenderStatistics

渲染信息统计信息参数，指示提交帧的帧率、渲染帧率、ATW渲染帧数目、渲染超时总帧数、渲染连续超时帧数目的结构体

成员名称	成员类型	成员说明
SubmitFrameRate	double	提交帧的帧率
RenderFrameRate	double	渲染帧率
ATWFrameCnt	double	ATW渲染帧数目
AvgRenderCost	double	渲染每帧平均耗时（暂不支持）
TimeoutFrameCnt	double	渲染超时总帧数
ContinousTimeoutCnt	double	渲染连续超时帧数目

4 公共接口类

4.1 HVRLayoutCore

HVRLayout类为SDK2.0提供相机组件。

概述

主要包含相机变换组件、左视图变换组件及右视图变换组件。

继承自：MonoBehaviour

公共静态属性

属性名称	属性类型	属性说明
m_CamCtrObj	Transform	相机变换组件
m_LeftCamObj	Transform	左视图变换组件
m_RightCamObj	Transform	右视图变换组件

4.1.1 m_CamCtrObj

Transform m_CamCtrObj

相机变换组件(Camera Transform Component)，用于获取HuaweiVR相机位置、旋转、缩放等信息。

示例

```
HVRLayoutCore.m_CamCtrObj.transform.position;   //获取相机位置信息
HVRLayoutCore.m_CamCtrObj.transform.rotation;    //获取相机旋转信息
//为中心相机添AudioListener脚本
HVRLayoutCore.m_CamCtrObj.gameObject.AddComponent<AudioListener>();
```

4.1.2 m_LeftCamObj

Transform m_LeftCamObj

左视图变换组件，可获取左视图位置，旋转，缩放等信息。

示例

```
//创建一个物体
GameObject go = new GameObject ();
//将物体挂在左眼相机上
```

```
go.transform.parent = HVRLayoutCore.m_LeftCamObj;
```

4.1.3 m_RightCamObj

Transform m_RightCamObj

右视图变换组件，可获取右视图位置，旋转，缩放等信息。

4.2 HVRCamCore

HVRCamCore类提供天空盒功能。

概述

主要提供使用天空盒的功能，通过替换天空盒材质，实现应用场景的自由设定。

继承自：MonoBehaviour

公共静态方法

方法名称	返回值类型	方法说明
UseSkyBox(bool bUse, Material mat = null)	bool	通过替换材质设定应用场景

4.2.1 UseSkyBox

bool UseSkyBox(bool bUse, Material mat = null)

是否使用天空盒场景，若使用，通过替换材质设定应用场景。

参数

参数名称	参数类型	参数说明
bUse	bool	bUse = true，绘制天空盒；bUse = false，不绘制天空盒
mat	Material	天空盒材质

返回值

绘制天空盒时，替换材质成功返回 true，失败返回 false；不绘制天空盒，返回 true。

示例

```
Material skyboxmat = Resources.Load ("Materials/Skybox") as Material;
if (skyboxmat != null) {
```

```
        bool ret = HVRCamCore.UseSkyBox (true, skyboxmat);
        if (ret){
             Debug.Log ("Materials load success!");
       } else{
            Debug.Log ("Materials load failed!");
       }
 } else {
       Debug.Log ("material not loaded");
}
```

4.3 HVRDefCore

HVRDefCore类提供安全模式图像颜色的公共枚举。

概述

主要提供设置安全模式后，显示图像颜色的类型。

4.3.1 SafeModeColor

SafeModeColor

指示设置安全模式后，显示图像颜色类型

枚举成员	说明
Default	默认颜色
White	白色
Red	红色
Green	绿色
Blue	蓝色
Yellow	黄色
Cyan	青色
Magenta	品红

4.4 HvrApi

SDK2.0 Unity API的主入口点。

概述

主要用于获取SDK2.0版本号，及各模块的句柄。

公共静态方法

方法名称	返回值类型	方法说明
GetHvrSdkVersion()	string	获取SDK2.0的版本号
GetHelmetHandle ()	IHelmetHandle	获取头盔句柄
GetControllerHandle ()	IControllerHandle	获取控制器句柄
GetRenderHandle()	IRenderHandle	获取显示句柄

4.4.1 GetHvrSdkVersion

string GetHvrSdkVersion()

获取SDK2.0版本号。

返回值

调用函数成功返回SDK2.0版本号字符串，失败返回空字符串。

示例

```
string version = HvrApi.GetHvrSdkVersion ();
Debug.Log ("SDK2.0 version :" + version);
```

4.4.2 GetHelmetHandle

IHelmetHandle GetHelmetHandle ()

获取头盔句柄IHelmetHandle，用于调用头盔相关功能的接口函数（请参考4.5节）。

返回值

调用函数成功返回头盔句柄IHelmetHandle，调用函数失败返回null。

备注

该接口需要在Unity生命周期的Start()或其之后调用。

示例

```
IHelmetHandle helmetHandle; //以下示例代码中涉及到的头盔句柄都用该变量
helmetHandle = HvrApi.GetHelmetHandle ();
 if (helmetHandle != null) {
       Debug.Log (" GetHelmetHandle Success! ");
} else {
        Debug.Log (" GetHelmetHandle Failed! ");
}
```

4.4.3 GetControllerHandle

IControllerHandle GetControllerHandle ()

获取控制器句柄IControllerHandle，用于调用控制器相关的接口函数（请参考4.6节）。

返回值

调用函数成功返回控制器句柄IControllerHandle，调用函数失败返回null。

备注

- 该接口需要在Unity生命周期的Start()或其之后调用。
- 如需在非脚本线程或应用主线程中调用该接口，需要在调用该接口前将当前线程绑定到Java线程，并在不需要调用该接口后解除绑定。

示例

```
IControllerHandle controllerHandle; //以下示例代码中涉及到的控制器都用该变量
 controllerHandle = HvrApi.GetControllerHandle ();
 if (controllerHandle != null) {
       Debug.Log (" GetControllerHandle Success! ");
} else {
      Debug.Log (" GetControllerHandle Failed! ");
 }
```

4.4.4 GetRenderHandle

IRenderHandle GetRenderHandle()

获取显示句柄，调用显示相关的功能函数（请参考4.8节）。

返回值

调用函数成功返回显示句柄IRenderHandle，失败返回null。

备注

该接口需要在Unity生命周期的Start()或其之后调用。

示例

```
IRenderHandle renderHandle; //以下示例代码中涉及到的渲染句柄都用该变量
 renderHandle = HvrApi.GetRenderHandle ();
 if (renderHandle != null) {
        Debug.Log ("GetRenderHandle Success!");
} else {
         Debug.Log ("GetRenderHandle Failed!");
```

```
}
```

4.4.5 EnableSvsEffect

int EnableSvsEffect (bool enable)

开启EMUI系统的SVS音效，该接口需要android.permission.MODIFY_AUDIO_SETTINGS权限。

开启后，应用中所有声音的方位会随着头转而发生变化，适用于VR中观看视频的场景，但开发者应注意开启的时机并及时关闭。

返回值

固定返回0。

示例

```
HvrApi.EnableSvsEffect (true);
```

4.5 IHelmetHandle

IHelmetHandle类提供头盔相关的功能。

概述

主要提供获取姿态、位置、传感器数据，锁定、解锁姿态等功能。

公共方法

方法名称	返回值类型	方法说明
IsAvailable()	bool	查询设备是否可用
GetPosture(ref Posture posture)	int	获取当前头盔的位置姿态
GetSensorInfo (ref SensorData sensor)	int	获取头盔传感器数据、预测时间
ResetCenter()	int	重置画面位置、姿态至初始位姿
ResetYaw ()	int	重置画面偏航角
ResetOrientation ()	int	重置陀螺仪传感器姿态
ResetPosition ()	int	重置当前位置至初始位置
SetPoseLock (bool enable)	int	开启关闭姿态锁定

SetPositionLock (bool enable)	int	开启关闭位置锁定
GetHelmetInfo(ref HelmetModel helmetModel)	int	获取头盔类型

4.5.1 IsAvailable

bool IsAvailable()

查询头盔设备是否可用，一般在获取到头盔句柄后调用。

返回值

设备可用返回true，不可用返回false。

示例

```
if (helmetHandle != null) {
    Debug.Log (" GetHelmetHandle Success! ");
    if (helmetHandle.IsAvailable ()) {
        Debug.Log (" Helmet is available! ");
    } else {
        Debug.Log (" Helmet is not available! ");
    }
} else {
    Debug.Log (" GetHelmetHandle Failed! ");
}
```

4.5.2 GetPosture

int GetPosture(ref Posture posture)

获取当前头盔的位置向量、跟踪状态和旋转四元数。

参数

参数名称	参数类型	参数说明
posture	Posture（参考3.1节介绍）	将当前头盔的旋转四元数(x,y,z,w)写入posture.rotation
		将当前头盔的位置向量(x,y,z）写入posture.position
		将当前头盔的跟踪状态(Tracking或Untracking)写入posture.trackStatus

返回值

调用函数成功返回0，失败返回-1。

示例

```
Posture pos = new Posture ();
 int ret = helmetHandle.GetPosture (ref pos);
 if (ret == 0) {
        Quaternion quatDat;
        Vector3 posDat;
        quatDat = pos.rotation; //获取旋转姿态信息
        posDat = pos.position; //获取位置信息
} else {
        Debug.Log (" Get VR glass posture failed! ");
}
```

4.5.3 GetSensorInfo

int GetSensorInfo (ref SensorData sensor)

获取当前头盔传感器数据、预测时间。

参数

参数名称	参数类型	参数说明
sensor	SensorData（参考3.2节介绍）	将头盔的陀螺仪数据(x,y,z)写入sensor.gypo
		将头盔的加速度计数据(x,y,z)写入sensor.accel
		将头盔的磁力计数据(x,y,z)写入sensor.mag
		将头盔的预测时间(单位：us)写入sensor.predictTime

返回值

调用函数成功返回0，失败返回-1。

示例

```
SensorData sensor = new SensorData();
int ret = helmetHandle.GetSensorInfo (ref sensor);
if( ret == 0 ){
        Vector3 gypoDat;
        gypoDat = sensor.gypo; //获取头盔陀螺仪数据
}else{
        Debug.Log(" Get VR glass SensorData failed! ");
}
```

4.5.4 ResetCenter

int ResetCenter()

将当前位置和姿态复位到初始位置和姿态。通常用于画面姿态、位置都偏离的情况（重置位置暂不支持）。

返回值

调用函数成功返回0，失败返回-1。

示例

```
//重置位置、姿态
if(helmetHandle.ResetCenter () == 0 ){
    Debug.Log(" Reset Center Success! ");
}else{
    Debug.Log(" Reset Center Failed! ");
}
```

4.5.5 ResetYaw

int ResetYaw ()

姿态重置偏航角。通常用于如下情况：当远离VR头盔，手机屏幕熄灭，再次佩戴头盔，点亮屏幕时，画面可能会发生视角偏离。

返回值

调用函数成功返回0，失败返回-1。

示例

```
//重置偏航角
if(helmetHandle.ResetYaw () == 0 ){
    Debug.Log(" Reset Yaw Success! ");
}else{
    Debug.Log(" Reset Yaw Failed! ");
}
```

4.5.6 ResetOrientation

int ResetOrientation ()

陀螺仪传感器姿态复位到初始姿态。通常用于画面姿态发生偏离的情况。

返回值

调用函数成功返回0，失败返回-1。

示例

```
//重置姿态
if(helmetHandle.ResetOrientation () == 0 ){
    Debug.Log(" Reset orientation Success! ");
}else{
    Debug.Log(" Reset orientation Failed! ");
}
```

4.5.7 ResetPosition（暂不支持）

int ResetPosition ()

将当前位置复位到初始位置。通常用于画面位置发生偏离的情况。

返回值

调用函数成功返回0，失败返回-1。

4.5.8 SetPoseLock

int SetPoseLock (bool enable)

开启关闭姿态锁定。通常用于观看电影的情况，观看者在调整观看姿势前后，均调用该接口，分别开启、关闭姿态锁定，以保证观看的画面姿态不会发生变化。

参数

参数名称	参数类型	参数说明
enable	bool	输入enable为true：开启姿态锁定；enable为false：解除姿态锁定

返回值

调用函数成功返回0，失败返回-1。

示例

```
if(helmetHandle.SetPoseLock (true) == 0 ){
    Debug.Log(" Set Pose Lock Success! " );
}else{
    Debug.Log(" Set Pose Lock Failed ! ");
```

```
}
if(helmetHandle.SetPoseLock (false) == 0 ){
    Debug.Log(" Set Pose Unlock Success ! " );
}else{
    Debug.Log(" Set Pose Unlock Failed! ");
}
```

4.5.9 SetPositionLock （暂不支持）

int SetPositionLock (bool enable)

开启关闭位置锁定。通常用于观看电影的情况，观看者在调整观看位置前后，均调用该接口，分别开启、关闭位置锁定，以保证观看的画面位置不会发生偏离。

参数

参数名称	参数类型	参数说明
enable	bool	输入enable为true：开启位置锁定；enable为false：解除位置锁定

返回值

调用函数成功返回0，失败返回-1。

4.5.10 GetHelmetInfo

int GetHelmetInfo(ref HelmetModel helmetModel)

获取头盔类型（HuaweiVR1.0或HuaweiVR2.0或其他头盔）。

参数

参数名称	参数类型	参数说明
helmetModel	HelmetModel（参考2.2节介绍）	将头盔的类型（HVR_HELMET_FIRST_GEN, HVR_HELMET_SECOND_GEN, HVR_HELMET_NOT_FOUND, HVR_HELMET_UNKNOWN ）写入helmetModel

返回值

调用函数成功返回0，失败返回-1。

示例

```
HelmetModel helmetmodel = HelmetModel.HVR_HELMET_UNKNOWN;
```

```
int ret = helmetHandle.GetHelmetInfo(ref helmetmodel);
if( ret == 0 ){
    switch(helmetmodel){
    case HelmetModel.HVR_HELMET_FIRST_GEN:
        Debug.Log("HuaweiVR1.0 helmet " );
        break;
    case HelmetModel.HVR_HELMET_SECOND_GEN:
        Debug.Log("HuaweiVR2.0 helmet " );
        break;
    case HelmetModel.HVR_HELMET_NOT_FOUND:
        Debug.Log(" not found helmet " );
        break;
    case HelmetModel.HVR_HELMET_UNKNOWN:
        Debug.Log(" unknow helmet " );
        break;
    }
}
```

4.6 IControllerHandle

IControllerHandle类提供控制器事件类、事件句柄、获取控制器对象、可用控制器的序列号等功能。

概述

主要提供监听控制器事件、获取控制器对象、获取可用控制器序号等功能。

事件类

事件类名	继承	说明
ControllerEventArgs	EventArgs	控制器上报事件类

4.6.1 ControllerEventArgs

控制器事件类，用于监听控制器断开、连接及低电量事件，包含控制器事件ID和事件数据两个属性

属性名称	属性类型	属性说明
eventId	ControllerEvent	控制器事件ID，指示控制器断开、连接及低电量事件（低电量事件暂不支持），参考2.4节

eventData	Object	控制器事件数据，指示连接、或断开的手柄序列号

事件句柄

句柄名称	句柄类型	句柄说明
ControllerStatusEventHandler	event EventHandler	控制器事件句柄

4.6.2 ControllerStatusEventHandler

event EventHandler ControllerStatusEventHandler

控制器事件句柄，通过注册控制器句柄来监听控制器断开、连接及低电量事件。

示例

```
void OnEnable ()
{
    controllerHandle.ControllerStatusEventHandler += Handle_Controller;
}
void Handle_Controller (object sender, EventArgs args)
{
    var controllerArgs = (ControllerEventArgs)args;
    ControllerEvent controllerEvent = (EventArgs)controllerArgs.eventId;
    int controllerIndex = (int)controllerArgs.eventData;
    switch (controllerEvent) {
    case ControllerEvent.ControllerEventConnected:
        Debug.Log (" Controller is connected " + controllerIndex);
        break;
    case ControllerEvent.ControllerEventDisconnected:
        Debug.Log (" Controller is disconnected " + controllerIndex);
        break;
    case ControllerEvent.ControllerEventLowPower:
        Debug.Log (" Controller is low power ");
        break;
    }
 }
```

公共方法

方法名称	返回值类型	方法说明
GetValidIndices ()	int[]	获取可用的控制器序列号
GetControllerByIndex (int	IController	获取序列号为index的控制器句

index)		柄
IsLeftHandMode()	bool	是否为左手模式

4.6.3 GetValidIndices

int[] GetValidIndices ()

获取可用的控制器序号（只是列出可用的控制器，不一定是已连接的），序号为 0 时，是标准输入，用于 HuaweiVR1.0 头盔触摸板，为 1、2…时，是手柄输入事件。

返回值

调用成功，返回可用控制器序号的数组，失败返回 null。

示例

```
if(null == controllerHandle){
    Debug.LogError("controllerHandle is null ");
}else{
    int[] indices = controllerHandle.GetValidIndices ();
}
```

4.6.4 GetControllerByIndex

IController GetControllerByIndex(int index)

获取序列号为 index 的控制器对象。

参数

参数名称	参数类型	参数说明
index	int	控制器的序列号

返回值

调用函数成功返回序列号为 index 的控制器对象，失败返回 null。

示例

```
private IController controller; //以下示例代码中涉及到的控制器变量都用该变量
void Start(){
    if(null == controllerHandle){
        Debug.LogError("controllerHandle is null ");
    }else{
        int[] indices = controllerHandle.GetValidIndices();
```

```
            // 获取手柄输入的控制器对象
            controller = controllerHandle.GetControllerByIndex(indices[1]);
        }
        if(null == controller){
            Debug.LogError(" Controller is null ");
        }
}
```

4.6.5 IsLeftHandMode

bool IsLeftHandMode()

是否启用左手模式，开发者需要在 OnApplicationPause(false)时调用该接口，根据是否是左手模式，调整虚拟手柄显示位置。

返回值

调用函数成功返回值为 true：是左手模式，false：不是左手模式。

示例

```
void OnApplicationPause(bool isPause){
    if( !isPause){
        if(null == controllerHandle){
            Debug.LogError("controllerHandle is null ");
        }else {
            if(controllerHandle.IsLeftHandMode()){
                Debug.LogError("controller is in leftHand mode! ");
            }
        }
    }
}
```

4.7 IController

IController类提供了控制器相关的功能。

概述

主要提供获取控制器状态、类型、电量、姿态位置、扳机数据、触摸板触摸位置、陀螺仪数据、加速度计数据，重置控制器姿态，控制器触摸板、按键动作判定，控制器震动等功能。

公共方法

方法名称	返回值类型	方法说明

IsAvailable()	bool	查询控制器设备是否可用
GetControllerStatus()	ControllerStatus	获取控制器当前状态
ResetCenter()	int	将当前位置、姿态重置为初始位置、姿态
GetControllerType()	ControllerType	获取控制器类型
GetBatteryLevel()	int	获取控制器电量
GetPosture(ref Posture pose)	int	获取控制器位置、姿态
GetTriggerData(ref float data)	int	获取控制器扳机数据
GetTouchpadTouchPos(ref Vector2 pos)	int	获取控制器触摸板触摸位置
GetGyroscope(ref Vector3 gyroscope)	int	获取控制器陀螺仪数据
GetAccelerometer (ref Vector3 accelerometer)	int	获取控制器加速度计数据
IsTouchpadTouching()	bool	判定触摸板当前状态是否被触摸
IsTouchpadTouchDown()	bool	判定触摸板是否触发按下动作
IsTouchpadTouchUp()	bool	判定触摸板是否触发抬起动作
IsTouchpadSwipeUp()	bool	判定触摸板是否触发上滑动作
IsTouchpadSwipeDown()	bool	判定触摸板是否触发下滑动作
IsTouchpadSwipeLeft()	bool	判定触摸板是否触发左滑动作
IsTouchpadSwipeRight()	bool	判定触摸板是否触发右滑动作

IsButtonPressed(ButtonType button)	bool	判定控制器按键当前状态是否被按下
IsButtonDown(ButtonType button)	bool	判定控制器按键是否触发按下动作
IsButtonUp(ButtonType button)	bool	判定控制器按键是否触发抬起动作
StartVibration()	int	使控制器开始震动
StopVibration()	int	使控制器停止震动

4.7.1 IsAvailable

bool IsAvailable ()

查询已连接的控制器是否可用。

返回值

返回true：可用，false：不可用。

示例

```
if(null == controller){
    Debug.LogError(" Controller is null ");
}else{
    if(controller.IsAvailable ()){
        Debug.Log (" Controller is available ");
    }
}
```

4.7.2 GetControllerStatus

ControllerStatus GetControllerStatus()

获取控制器当前状态。

返回值

返回控制器当前连接状态，控制器状态类型ControllerStatus请参考2.5节介绍。

示例

```
if(null == controller){
    Debug.LogError(" Controller is null ");
```

```
}else{
      ControllerStatus status = controller. GetControllerStatus();
       switch (status) {
       case ControllerStatus.ControllerStatusDisconnected:
            Debug.Log (" Controller is disconnected ");
            break;
       case ControllerStatus.ControllerStatusScanning:
            Debug.Log (" Controller is scanning ");
            break;
       case ControllerStatus.ControllerStatusConnecting:
            Debug.Log (" Controller is connecting ");
            break;
       case ControllerStatus.ControllerStatusConnected:
            Debug.Log (" Controller is connected ");
            break;
       case ControllerStatus.ControllerStatusError:
            Debug.Log (" Controller is connect error ");
            break;
       }
}
```

4.7.3 ResetCenter（暂不支持）

int ResetCenter()

将当前位置和姿态复位到初始位置和姿态。通常用于画面姿态、位置都偏离的情况（重置位置暂不支持）。

返回值

调用函数成功返回0，失败返回-1。

示例

```
if(null == controller){
      Debug.LogError(" Controller is null ");
}else{
      controller. ResetCenter();
}
```

4.7.4 GetControllerType（暂不支持）

ControllerType GetControllerType()

获取控制器类型（3Dof、6Dof、游戏、标准输入等）。

返回值

调用函数成功返回当前连接的控制器类型，控制器类型ControllerType请参考2.3节介绍。

示例

```
if(null == controller){
    Debug.LogError(" Controller is null ");
}else{
    controllerType type = Controller.GetControllerType ();
}
```

4.7.5 GetBatteryLevel（暂不支持）

int GetBatteryLevel()

获取控制器电量值。

返回值

调用函数成功返回控制器电量值，失败返回-1。

示例

```
if(null == controller){
    Debug.LogError(" Controller is null ");
}else{
    int batteryLevel = controller.GetBatteryLevel();
}
```

4.7.6 GetPosture

int GetPosture(ref Posture pose)

获取当前控制器的位置向量、跟踪状态和旋转四元数（获取位置向量暂不支持）。

参数

参数名称	参数类型	参数说明
pose	Posture（参考3.1节介绍）	将当前控制器的旋转四元数(x,y,z,w)写入pose.rotation
		将当前控制器的位置向量(x,y,z）写入pose.position
		将当前控制器的跟踪状态(Tracking或Untracking)写入pose.trackStatus

返回

调用函数成功返回0，失败返回-1。

示例

```
if(null == controller){
    Debug.LogError(" Controller is null ");
}else{
    Posture pos = new Posture ();
    int ret = controller.GetPosture(ref pos);
    if( ret == 0 ){
        Quaternion quatDat;
        Vector3 posDat;
        quatDat = pos.rotation; //获取手柄旋转姿态信息
        posDat = pos.position; //获取手柄位置信息
    }else{
        Debug.Log(" Get controller posture failed! ");
    }
}
```

4.7.7 GetTriggerData

int GetTriggerData(ref float data)

获取当前控制器的扳机数据。

参数

参数名称	参数类型	参数说明
data	float	将当前控制器的扳机数据(取值为[0.0f, 1.0f]的浮点数)写入data

返回值

调用函数成功返回0，失败返回-1。

示例

```
if(null == controller){
Debug.LogError(" Controller is null ");
}else{
    float triggerData = 0.0f;
    int ret = controller.GetTriggerData(ref triggerData);
    if( ret == 0 ){
        Debug.Log(" Get controller trigger data success! ");
     }else{
        Debug.Log(" Get controller trigger data failed! ");
```

```
    }
}
```

备注

当前版本的控制器扳机数据仅为0或1。

4.7.8 GetTouchpadTouchPos

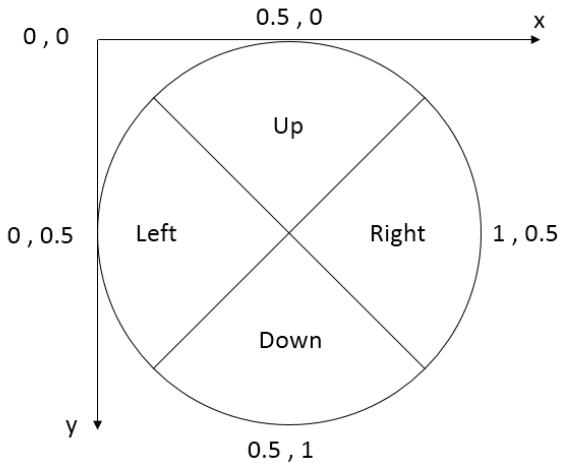
int GetTouchpadTouchPos(ref Vector2 pos)

获取当前控制器触摸板的触摸位置向量。

参数

参数名称	参数类型	参数说明
pos	Vector2	将当前控制器的位置向量(x,y)写入pos

手柄触摸板获取到的位置向量的坐标系如下所示，使用时需根据实际坐标系作转换：



返回值

调用函数成功返回0，失败返回-1。

示例

```
if(null == controller){
    Debug.LogError(" Controller is null ");
}else{
    Vector2 touchpdPos = new Vector2();
    int ret = controller.GetTouchpadTouchPos(ref touchpdPos);
     if( ret == 0 ){
          Debug.Log(" Get controller touchpdPos success! ");
```

```
    }else{
        Debug.Log(" Get controller touchpdPos failed! ");
    }
}
```

4.7.9 GetGyroscope（暂不支持）

int GetGyroscope(ref Vector3 gyroscope)

获取当前控制器的陀螺仪数据。

参数

参数名称	参数类型	参数说明
gyroscope	Vector3	将当前控制器陀螺仪的位置向量(x,y,z)写入gyroscope

返回值

调用函数成功返回0，失败返回-1。

示例

```
if(null == controller){
    Debug.LogError(" Controller is null ");
}else{
    Vector3 gyroscope = new Vector3();
    int ret = controller.GetGyroscope(ref gyroscope);
    if( ret == 0 ){
        Debug.Log(" Get controller gyroscope success! ");
    }else{
        Debug.Log(" Get controller gyroscope failed! ");
    }
}
```

4.7.10 GetAccelerometer（暂不支持）

int GetAccelerometer(ref Vector3 accelerometer)

获取当前控制器的加速度计数据。

参数

参数名称	参数类型	参数说明
accelerometer	Vector3	将当前控制器加速度计的位置向量(x,y,z)写入accelerometer

返回值

调用函数成功返回0，失败返回-1。

示例

```
if(null == controller){
    Debug.LogError(" Controller is null ");
}else{
    Vector3 accelerometer = new Vector3();
    int ret = controller. GetAccelerometer(ref accelerometer);
    if( ret == 0 ){
        Debug.Log(" Get controller accelerometer success! ");
    }else{
        Debug.Log(" Get controller accelerometer failed! ");
    }
}
```

4.7.11 IsTouchpadTouching

bool IsTouchpadTouching()

判定触摸板当前状态是否被触摸。

返回值

触摸板状态正在被触摸返回true，否则返回false。

4.7.12 IsTouchpadTouchDown

bool IsTouchpadTouchDown()

判定触摸板是否触发按下动作，兼容HuaweiVR1.0的触摸板触摸按下功能。

返回值

触摸板触发按下动作则返回true，否则返回false。

4.7.13 IsTouchpadTouchUp

bool IsTouchpadTouchUp()

判定触摸板是否触发抬起动作，兼容HuaweiVR1.0的触摸板触摸抬起功能。

返回值

触摸板触发抬起动作则返回true，否则返回false。

4.7.14 IsTouchpadSwipeUp

bool IsTouchpadSwipeUp ()

判定触摸板是否触发上滑动作，兼容HuaweiVR1.0的触摸板向上滑动功能。

返回值

触摸板触发上滑动作则返回true，否则返回false。

4.7.15 IsTouchpadSwipeDown

bool IsTouchpadSwipeDown ()

判定触摸板是否触发下滑动作，兼容HuaweiVR1.0的触摸板向下滑动功能。

返回值

触摸板触发下滑动作则返回true，否则返回false。

4.7.16 IsTouchpadSwipeLeft

bool IsTouchpadSwipeLeft ()

判定触摸板是否触发左滑动作，兼容HuaweiVR1.0的触摸板向左滑动功能。

返回值

触摸板触发左滑动作则返回true，否则返回false。

4.7.17 IsTouchpadSwipeRight

bool IsTouchpadSwipeRight ()

判定触摸板是否触发右滑动作，兼容HuaweiVR1.0的触摸板向右滑动功能。

返回值

触摸板触发右滑动作则返回true，否则返回false。

示例

```
void Start(){
    if(null == controller){
        Debug.LogError(" Controller is null ");
    }
}
```

```
//具体接口调用位置，按照开发者自己的需求
void Update(){
    if (controller. IsTouchpadTouching ()) {
        Debug.Log (" Touchpad is touching ");
    }
    if (controller.IsTouchpadSwipeDown ()) {
        Debug.Log (" Touchpad is swipe down ");
    }else if(controller.IsTouchpadSwipeUp()){
        Debug.Log ("Touchpad is swipe up ");
    }else if(controller.IsTouchpadSwipeLeft()){
        Debug.Log ("Touchpad is swipe left ");
    }else if(controller.IsTouchpadSwipeRight()){
        Debug.Log ("Touchpad is swipe right ");
    }
    if(controller.IsTouchpadTouchUp()){
        Debug.Log (" Touchpad is touch Up ");
    }else if(controller.IsTouchpadTouchDown()){
        Debug.Log (" Touchpad is touch Down ");
    }
}
```

4.7.18 IsButtonPressed

bool IsButtonPressed(ButtonType button)

判定控制器按键当前状态是否被按下。

参数

参数名称	参数类型	参数说明
button	ButtonType（参考2.6节介绍）	控制器按键的类型

返回值

对应按键类型的按键当前状态被按下则返回true，否则返回false。

4.7.19 IsButtonDown

bool IsButtonDown(ButtonType button)

判定控制器按键是否触发按下动作，当ButtonType为ButtonBack时，兼容HuaweiVR1.0的返回按键按下功能。

参数

参数名称	参数类型	参数说明
button	ButtonType（参考2.6节介绍）	控制器按键的类型

返回值

对应按键类型的按键触发按下动作则返回true，否则返回false。

备注

当ButtonType为ButtonBack时，在HuaweiVR2.0中，不能调用此接口响应需要按返回键来执行的事件（如，退出事件等），可以调用IsButtonUp(ButtonType button)接口响应。

4.7.20 IsButtonUp

bool IsButtonUp(ButtonType button)

判定控制器按键是否触发抬起动作，当ButtonType为ButtonConfirm时，兼容HuaweiVR1.0的触摸板单击功能；当ButtonType为ButtonBack时，兼容HuaweiVR1.0的返回按键抬起功能。

参数

参数名称	参数类型	参数说明
button	ButtonType（参考2.6节介绍）	控制器按键的类型

返回值

对应按键类型的按键触发抬起动作则返回true，否则返回false。

示例

```
if(null == controller){
    Debug.LogError(" Controller is null ");
}else{
    if(controller.IsButtonPressed (ButtonType.ButtonConfirm)){
        Debug.Log(" Confirm button is pressed! ");
    }
    if(controller.IsButtonDown (ButtonType.ButtonConfirm)){
        Debug.Log(" Confirm button is pressed down! ");
    }
    if(controller.IsButtonUp (ButtonType.ButtonConfirm)){
        Debug.Log(" Confirm button is pressed up! ");
```

```
    }
}
```

备注

- 控制器上的ButtonHome、ButtonVolumeInc、ButtonVolumeDec按键为预置功能按键，为避免使用冲突，普通应用获取其值时总返回false。
- 当ButtonType为ButtonBack时，在HuaweiVR2.0中，调用此接口响应需要按返回键来执行的事件（如，退出事件等）。

4.7.21 StartVibration()（暂不支持）

int StartVibration()

使控制器开始震动。

返回值

调用函数成功返回0，失败返回-1。

示例

```
if(null == controller){
    Debug.LogError(" Controller is null ");
}else{
    int ret = controller.StartVibration();
    if(ret == 0){
        Debug.Log(" Controller start vibration! ");
    }
}
```

4.7.22 StopVibration()（暂不支持）

int StopVibration()

使控制器停止震动。

返回值

调用函数成功返回 0，失败返回-1。

示例

```
if(null == controller){
    Debug.LogError(" Controller is null ");
}else{
```

```
    int ret = controller.StopVibration ();
    if(ret == 0){
        Debug.Log(" Controller stop vibration! ");
    }
}
```

4.8 IRenderHandle

IRenderHandle类提供与显示相关的功能。

概述

主要用于捕获应用输入的图像，使能色散等显示相关功能。

公共方法

方法名称	返回值类型	方法说明
CaptureEyeImage(String filePath)	int	捕获应用输入的图像
EnableChromaticAberration(bool enable)	void	开启或关闭色散功能
GetRenderStatics(ref RenderStatistics renderStatics)	int	获取显示统计信息

4.8.1 CaptureEyeImage

int CaptureEyeImage(string filePath)

捕获应用输入的图像（输出为JPEG格式的图像），存放至输入路径：filePath下(如：/sdcard/captures/cp.jpeg)下。支持连续捕获（每帧调用一次）。

参数

参数名称	参数类型	参数说明
filePath	string	输入图像保存路径

返回值

调用函数成功返回0，失败返回-1。

备注

- 如需在非脚本线程或应用主线程中调用该接口，需要在调用该接口前将当前线程绑定到Java线程，并在不需要调用该接口后解除绑定。

- 使用截图功能时，在Unity场景需进行读写权限设置，如下：

  Player Settings… -> Other Settings -> Write Permission选择“External（SDCard）”

示例

```
int i = 0;
void Update(){
    i.ToString();
    string filePath = "/sdcard/captures/" + i + ".jpeg";
    int ret = renderHandle.CaptureEyeImage (filePath);
    i++;
    if(ret == 0){
        Debug.Log(" Capture Images Success! ");
    }else{
        Debug.Log(" Capture Images Failed! ");
    }
}
```

4.8.2 EnableChromaticAberration

void EnableChromaticAberration(bool enable)

开启或关闭色散功能，enable为true：开启色散功能，enable为false:关闭色散功能。

参数

参数名称	参数类型	参数说明
enable	bool	使能色散功能的输入参数（true：开启色散功能， false:关闭色散功能）

备注

该接口需在Updte()中每帧都调用一次进行设置反色散开启功能，或者在Start()或其他接口中延迟四帧调用该接口。

示例

```
void Update(){
    renderHandle.EnableChromaticAberration(true);
}
```

4.8.3 GetRenderStatics

int GetRenderStatics(ref RenderStatistics renderStatics)

获取渲染相关统计信息（提交帧率、渲染帧率、渲染超时帧数等信息）。

参数

参数名称	参数类型	参数说明
renderStatics	RenderStatistics （参考3.3节介绍）	渲染统计信息

返回值

调用函数成功返回0，失败返回-1。

示例

```
RenderStatistics renderStatics;
void Update(){
    renderHandle.GetRenderStatics (ref renderStatics);
    int submiteFrameRate = (int)renderStatics.SubmitFrameRate; //提交帧率
    int renderFrameRate = (int)renderStatics.RenderFrameRate; //渲染帧率
}
```

4.8.4 EnableMonocularDisplay

void EnableMonocularDisplay(bool enable)

开启单目渲染功能，enable为true：单目渲染模式，enable为false：普通双目渲染模式。

参数

参数名称	参数类型	参数说明
enable	bool	开启单目渲染功能的输入参数（true： 单目渲染模式， false： 普通双目渲染模式）

备注

- 调用时序：需要在Update()前调用该接口；
- 单目渲染模式无视差效果，请开发者根据场景判断是否需要开启该模式；
- 调用该接口后，后续场景均会持续当前渲染模式。

示例

```
void Start(){
    renderHandle.EnableMonocularDisplay (true);  //开启单目渲染模式
}
```

4.9 HVRPluginCore

HVRPluginCore类设置提供远、近裁剪平面距离及安全模式相关功能。

概述

主要用于提设置供远、近裁剪平面距离，开启、关闭安全模式及设置安全模式开启时间等功能。

公共静态方法

方法名称	返回值类型	方法说明
HVR_SetClipPlaneParams(float m_nearClipPlane = 0.01f, float m_farClipPlane = 1000.0f)	void	设置远、近裁剪平面距离
HVR_OpenSafeMode()	void	开启安全模式
HVR_CloseSafeMode()	void	关闭安全模式
HVR_SetSafeModeParams(float durationSeconds)	void	设置安全模式开始的时间长度
HVR_SetSafeModeParams(byte mAlpha = 64, HVRDefCore.SafeModeColor mColor = HVRDefCore.SafeModeColor.Default, HVRDefCore.SafeModeColor mCentralColor = HVRDefCore.SafeModeColor.Default)	void	设置安全模式开启后的图像透明度和颜色

4.9.1 HVR_SetClipPlaneParams

void HVR_SetClipPlaneParams(float m_nearClipPlane, float m_farClipPlane)

动态设置近裁剪平面距离和远裁剪平面距离。

参数说明

参数名称	参数类型	参数说明
m_nearClipPlane	float	近裁剪平面距离（默认为0.01）
m_farClipPlane	float	远裁剪平面距离（默认为1000）

示例

```
float mNearClipPlane = 0.02f;
float mFarClipPlane=1200.0f;
HVRPluginCore.HVR_SetClipPlaneParams(mNearClipPlane, mFarClipPlane);
```

4.9.2 HVR_OpenSafeMode

void HVR_OpenSafeMode()

开启安全模式，打开手机摄像头，并将手机摄像头采集的数据信息映射到VR场景中，对HuaweiVR1.0头盔用户进行外界环境的提示。

示例

```
HVRPluginCore.HVR_OpenSafeMode();
```

备注

仅支持HuaweiVR 1.0设备

4.9.3 HVR_CloseSafeMode

void HVR_CloseSafeMode ()

关闭安全模式，关闭手机摄像头，取消VR场景中映射手机摄像头采集的数据信息，取消对HuaweiVR1.0头盔用户进行外界环境的提示。

示例

```
HVRPluginCore.HVR_CloseSafeMode;
```

备注

仅支持HuaweiVR 1.0设备

4.9.4 HVR_SetSafeModeParams

- **void HVR_SetSafeModeParams(float durationSeconds)**

设置安全模式开启时间长度，以开启安全模式函数调用成功为起始时间点，经过所设置的时间长度后，自动关闭安全模式，以保证摄像头不会因长时开启而影响用户使用设备，若设置时间长度小于等于0，则打开安全模式后将不会关闭，建议在调用开启安全模式函数前调用本函数。

开启安全模式时间默认10s，建议开发者对这个接口的输入参数设置值大小不要超过10s，

若长时开启安全模式，会增加应用运行功耗。

参数

参数名称	参数类型	参数说明
durationSeconds	float	安全模式开启时间长度，单位秒

示例

```
float durationSeconds = 10.0f;  //设置安全模式开启10s后自动关闭安全模式
HVRPluginCore.HVR_SetSafeModeParams (durationSeconds);
```

备注

仅支持HuaweiVR 1.0设备

- **HVR_SetSafeModeParams(byte mAlpha = 64, HVRDefCore.SafeModeColor mColor, HVRDefCore.SafeModeColor mCentralColor)**

设置开启安全模式后显示图像的透明度和颜色，可改变透明度适当值，不影响场景可视度，同时可分别改变安全模式中间区域的颜色和周围的颜色，这里的中间区域指用户需要特别关注的区域。

参数

参数名称	参数类型	参数说明
mAlpha	byte	图像透明度，0表示完全透明，这里限制最小值为16，255表示不透明（默认 64）
mColor	HVRDefCore.SafeModeColor	图像颜色（参考4.3.1）
mCentralColor	HVRDefCore.SafeModeColor	图像中间区域颜色（参考4.3.1）

示例

```
byte mAlpha = 64;
HVRDefCore.SafeModeColor mColor = HVRDefCore.SafeModeColor.White;
HVRDefCore.SafeModeColor mCentralColor = HVRDefCore.SafeModeColor.Yellow;
HVRPluginCore. HVR_SetSafeModeParams(mAlpha, mColor, mCentralColor);
```

备注

仅支持HuaweiVR 1.0设备

4.10 HvrEventListner

HvrEventListner提供静态委托变量，用来传递开发者想要监听的头瞄注视到的UI输入事件（包含：点击、按下、抬起、进入、退出事件）

公共静态委托变量

委托变量名	委托变量类型	委托变量说明
onEnter	VoidDelegate	响应进入事件
onExit	VoidDelegate	响应退出事件
onDown	VoidDelegate	响应按下事件
onUp	VoidDelegate	响应抬起事件
onClick	VoidDelegate	响应点击事件

委托变量类型 VoidDelegate

public delegate void VoidDelegate (GameObject go)

备注

仅支持HuaweiVR 1.0设备

示例

```
//头瞄注视退出按钮时，点击头盔触摸板，应用退出（该脚本绑定到退出按钮上）：
void Start () {
    HvrEventListner.Get(transform.gameObject).onClick = onPointClick;
}
private void onPointClick(GameObject go){
    if(go == transform.gameObject){
        Application.Quit ();
    }
}
```