1 HVRCORE

HVRCORE命名空间提供HUAWEI VR SDK for Unity（以下简称HVR SDK）的公共枚举、公共结构体、公共接口类等。

1.1 Classes

Class name	Class description
HVRLayout	*为HVR SDK提供相机组件*
HVRCamCore	*提供天空盒功能*
HvrApi	*用于获取HVR SDK版本号，及各模块的句柄*
HVRPluginCore	*提供设置远、近裁剪平面距离的功能*
ControllerEventArgs	*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*
HVREventListener	*提供委托变量，用来传递开发者想要监听的射线指向的UI输入事件（包含：点击、按下、抬起、进入、退出事件）*
HVRHelpMessage	*提供静态变量，用于自定义手柄帮助信息文字、字体颜色、字体大小、背景板颜色等*
HVRArmModel	*提供静态变量，用于自定义3Dof手柄模拟6Dof时手柄的初始位置及肘距*
HVRLinePointer	*提供与手柄射线光标相关的功能*

1.2 Interfaces

Interface name	Interface description
IHelmetHandle	*提供获取姿态、位置、传感器数据，锁定、解锁姿态等功能*
IControllerHandle	*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*
IController	*提供了控制器相关的功能*
IRenderHandle	*提供与显示相关的功能*

1.3 Enums

Enum name	Enum description
TrackStatus	*头盔跟踪状态*
HelmetModel	*头盔类型*
ControllerType	*控制器类型*
ControllerEvent	*控制器连接事件*
ControllerStatus	*控制器连接状态*
ButtonType	*控制器按钮类型*

1.3.1 TrackStatus

Enum
public enum TrackStatus { *Tracking = 0,* *Untracking = 1* *}*
头盔跟踪状态，指示头部跟踪数据是否有效

Enum Values	description
Tracking	*表示头盔跟踪数据有效*
Untracking	*表示头盔跟踪数据无效*

1.3.2 HelmetModel

Enum
public enum HelmetModel { *HVR_HELMET_FIRST_GEN,* *HVR_HELMET_SECOND_GEN,* *HVR_HELMET_THIRD_GEN,* *HVR_HELMET_FOURTH_GEN,* *HVR_HELMET_NOT_FOUND = 0xFFFE,* *HVR_HELMET_UNKNOWN = 0xFFFF* *}*
头盔类型，指示HUAWEI VR头盔类型

Enum Values	description
HVR_HELMET_FIRST_GEN	*HUAWEI VR 1.0头盔*
HVR_HELMET_SECOND_GEN	*HUAWEI VR 2*
HVR_HELMET_THIRD_GEN	*HUAWEI VR Glass*
HVR_HELMET_NOT_FOUND	*未找到头盔*
HVR_HELMET_UNKNOWN	*未知类型*

1.3.3 ControllerType

Enum
public enum ControllerType { *Controller3DOF,* *Controller6DOF,* *ControllerGaming,* *ControllerSysStd,* *ControllerOther* *}*
控制器类型，指示手柄的类型

Enum Values	description
Controller3DOF	*3Dof型*
Controller6DOF	*6Dof型*
ControllerGaming	*游戏型*
ControllerSysStd	*系统标准输入型*
ControllerOther	*其他类型*

1.3.4 ControllerEvent

Enum
public enum ControllerEvent { *ControllerEventConnected,* *ControllerEventDisconnected,* *ControllerEventLowPower,* *ControllerEventReset* *}*

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

Enum Values	description
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*
ControllerEventReset	*手柄位姿重置事件*

1.3.5 ControllerStatus

Enum
public enum ControllerStatus{ *ControllerStatusDisconnected,* *ControllerStatusScanning,* *ControllerStatusConnecting,* *ControllerStatusConnected,* *ControllerStatusError* *}*
控制器连接状态，主动查询控制器当前状态

Enum Values	description
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*

1.3.6 ButtonType

Enum
public enum ButtonType { *ButtonHome,* *ButtonBack,* *ButtonVolumeInc,* *ButtonVolumeDec,*

ButtonConfirm, *ButtonTrigger,* *ButtonTouchPad* *}*
控制器按键类型，指示控制器包含的所有按键类型

Enum Values	description
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*

HVR 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	*主页键Home*	*<1s*	*进入VRLauncher界面*
长按	*主页键Home*	*>=1s*	*视点重置*
长按	*返回键*	*>=3s*	*进入VRLauncher界面*
同时按下	*主页键Home & 扳机键*	*<1s*	*实现截图*

TIPS:

- 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）”

1.4 Structures

name	description
Posture	*姿态、位置参数及跟踪状态*
SensorData	*头盔传感器参数*
RenderStatistics	*显示统计参数*

1.4.1 Posture

Enum
public struct Posture{ *public Quaternion rotation;* *public Vector3 position;* *public TrackStatus trackStatus;* *}*
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

Structure Variables	description
rotation	*旋转四元数（x,y,z,w）*
position	*位置向量（x,y,z）*
trackStatus	*头部跟踪状态*

1.4.2 SensorData

Enum
public struct SensorData{ *public Vector3 gypo;* *public Vector3 accel;* *public Vector3 mag;* *public ulong predictTime;* *}*
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

Structure Variables	description
gypo	*陀螺仪参数*
accel	*加速度计参数*
mag	*磁力计参数*
predictTime	*预测时间（us）*

1.4.3 RenderStatistics

Enum
public struct RenderStatistics{ *public double SubmitFrameRate;* *public double RenderFrameRate;* *public double ATWFrameCnt;* *public double AvgRenderCost;* *public double TimeoutFrameCnt;* *public double ContinousTimeoutCnt;* *}*
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

Structure Variables	description
SubmitFrameRate	*提交帧的帧率*
RenderFrameRate	*渲染帧率*
ATWFrameCnt	*ATW渲染帧数目*
AvgRenderCost	*渲染每帧平均耗时（暂不支持）*
TimeoutFrameCnt	*渲染超时总帧数*
ContinousTimeoutCnt	*渲染连续超时帧数目*

2 HVRLayoutCore

CLASS info
public class HVRLayoutCore : MonoBehaviour
提供相机变换组件、左视图变换组件及右视图变换组件

2.1 Public Static Variable Summary

Variable Type	Variable name
Transform	*m_CamCtrObj*
Transform	*m_LeftCamObj*
Transform	*m_RightCamObj*

2.2 Public Static Variables

2.2.1 m_CamCtrObj

Variable
public static Transform m_CamCtrObj
相机变换组件(Camera Transform Component)，用于获取HUAWEI VR相机位置、旋转、缩放等信息

Code Example

```
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>();
```

2.2.2 m_LeftCamObj

Variable
public static 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

Code Example

```
//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;
```

2.2.3 m_RightCamObj

Variable
public static 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

3 HVRCamCore

CLASS info
public class HVRCamCore: MonoBehaviour
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

3.1 Public Static Methord Summary

Return type	Method name
bool	*UseSkyBox(bool bUse, Material mat = null)*

3.2 Public Static Methords

3.2.1 UseSkyBox

Method
public static 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

Parameters

Parameter name	Parameter desc
bUse	*bUse = true，绘制天空盒；bUse = false，不绘制天空盒*
mat	*天空盒材质*

Return

type	desc
bool	*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*

Code Example

```
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 HvrApi

CLASS info
public class HvrApi

用于获取HVR SDK版本号，及各模块的句柄

4.1 Public Static Methord Summary

Return type	Method name
string	*GetHvrSdkVersion()*
IHelmetHandle	*GetHelmetHandle ()*
IControllerHandle	*GetControllerHandle ()*
IRenderHandle	*GetRenderHandle()*

4.2 Public Static Methords

4.2.1 GetHvrSdkVersion

Method
public static string GetHvrSdkVersion()
获取HVR SDK版本号

Return

type	desc
string	*调用函数成功返回HVR SDK版本号字符串，失败返回空字符串*

Code Example

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

4.2.2 GetHelmetHandle

Method
public static IHelmetHandle GetHelmetHandle ()
获取头盔句柄IHelmetHandle，用于调用头盔相关功能的接口函数（请参考第7节）

Return

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

TIPS: 该接口需要在Unity生命周期的Start()或其之后调用。

Code Example

```
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! ");
```

4.2.3 GetControllerHandle

Method
public static IControllerHandle GetControllerHandle ()
获取控制器句柄IControllerHandle，用于调用控制器相关的接口函数（请参考第8节）

Return

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

TIPS:

- 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。

Code Example

```
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! ");
 }
```

4.2.4 GetRenderHandle

Method
public static IRenderHandle GetRenderHandle ()
获取显示句柄，调用显示相关的功能函数（请参考第10节）

Return

type	desc

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

TIPS: 该接口需要在Unity生命周期的Start()或其之后调用。

Code Example

```
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 ();
```

5 HVRPluginCore

CLASS info
public class HVRPluginCore
提供设置远、近裁剪平面距离的功能

5.1 Public Static Methord Summary

Return type	Method name
void	*HVR_SetClipPlaneParams(float m_nearClipPlane = 0.01f, float m_farClipPlane = 1000.0f)*

5.2 Public Static Methords

5.2.1 SetClipPlaneParams

Method
public static 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

Parameters

Parameter name	Parameter desc
m_nearClipPlane	*近裁剪平面距离（默认为0.01）*
m_farClipPlane	*远裁剪平面距离（默认为1000）*

Code Example

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

6 ControllerEventArgs

CLASS info
public class ControllerEventArgs : EventArgs
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和事件数据两个公共变量

6.1 Public Variable Summary

Variable Type	Variable name
ControllerEvent	*eventId*
Object	*eventData*

6.2 Public Variables

6.2.1 eventId

Variable
public ControllerEvent eventId
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1.3.4节

6.2.2 eventData

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

7 IHelmetHandle

interface info
public interface IHelmetHandle
提供获取姿态、位置、传感器数据，锁定、解锁姿态等功能

7.1 Public Methord Summary

Return type	Method name
bool	*IsAvailable()*

int	*GetPosture(ref Posture posture)*
int	*GetSensorInfo(ref SensorData sensor)*
int	*ResetCenter()*
int	*ResetYaw()*
int	*ResetOrientation()*
int	*ResetPosition()*
int	*SetPoseLock(bool enable)*
int	*SetPositionLock(bool enable)*
int	*GetHelmetInfo(ref HelmetModel helmetModel)*

7.2 Public Methords

7.2.1 IsAvailable

Method
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

Return

type	desc
bool	*设备可用返回true，不可用返回false*

Code Example

```
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! ");
}
```

7.2.2 GetPosture

Method
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

Parameters

Parameter name	Parameter desc
posture	*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

Return

type	desc
int	*调用函数成功返回0，失败返回-1*

Code Example

```
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! ");
}
```

7.2.3 GetSensorInfo

Method
int GetSensorInfo (ref SensorData sensor)
获取当前头盔传感器数据、预测时间

Parameters

Parameter name	Parameter desc
sensor	*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

Return

type	desc
int	*调用函数成功返回0，失败返回-1*

Code Example

```
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! ");
}
```

7.2.4 ResetCenter

Method
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）

Return

type	desc
int	*调用函数成功返回0，失败返回-1*

Code Example

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

7.2.5 ResetYaw

Method
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

Return

type	desc
int	*调用函数成功返回0，失败返回-1*

Code Example

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

7.2.6 ResetOrientation

Method
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

Return

type	desc
int	*调用函数成功返回0，失败返回-1*

Code Example

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

7.2.7 ResetPosition（暂不支持）

Method
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

Return

type	desc
int	*调用函数成功返回0，失败返回-1*

7.2.8 SetPoseLock

Method
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

Parameters

Parameter name	Parameter desc
enable	*输入enable为true：开启姿态锁定；enable为false：解除姿态锁定*

Return

type	desc
int	*调用函数成功返回0，失败返回-1*

Code Example

```
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! ");
}
```

7.2.9 SetPositionLock（暂不支持）

Method
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

Parameters

Parameter name	Parameter desc
enable	*输入enable为true：开启位置锁定；enable为false：解除位置锁定*

Return

type	desc
int	*调用函数成功返回0，失败返回-1*

7.2.10 GetHelmetInfo

Method
int GetHelmetInfo(ref HelmetModel helmetModel)
获取头盔类型（HUAWEI VR Glass或HUAWEI VR 2或HUAWEI VR 1.0或其他头盔）

Parameters

Parameter name	Parameter desc
helmetModel	*将头盔的类型（HVR_HELMET_FIRST_GEN, HVR_HELMET_SECOND_GEN, HVR_HELMET_THIRD_GEN， HVR_HELMET_FOURTH_GEN， HVR_HELMET_NOT_FOUND, HVR_HELMET_UNKNOWN ）写入helmetModel*

Return

type	desc
int	*调用函数成功返回0，失败返回-1*

Code Example

```
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("Huawei VR 1.0 helmet " );
        break;
    case HelmetModel.HVR_HELMET_SECOND_GEN:
        Debug.Log("Huawei VR 2" );
        break;
    case HelmetModel.HVR_HELMET_ THIRD _GEN:
        Debug.Log("Huawei VR Glass " );
        break;
    case HelmetModel.HVR_HELMET_NOT_FOUND:
        Debug.Log(" not found helmet " );
        break;
    case HelmetModel.HVR_HELMET_UNKNOWN:
```

```
            Debug.Log(" unknow helmet " );
            break;
        }
}
```

8 IControllerHandle

interface info
public interface IControllerHandle
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

8.1 Public Methord Summary

Return type	Method name
event EventHandler	*ControllerStatusEventHandler*
int[]	*GetValidIndices ()*
IController	*GetControllerByIndex (int index)*
bool	*IsLeftHandMode()*

8.2 Public Methords

8.2.1 ControllerStatusEventHandler

Method
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

Code Example

```
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;
    }
}
```

8.2.2 GetValidIndices

Method
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HUAWEI VR 1.0头盔触摸板，为1、2...时，是手柄输入事件

Return

type	desc
int[]	*调用成功，返回可用控制器序号的数组，失败返回null*

Code Example

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

8.2.3 GetControllerByIndex

Method
IController GetControllerByIndex(int index)
获取序列号为index的控制器对象

Parameters

Parameter name	Parameter desc
index	*控制器的序列号*

Return

type	desc

IController	*调用函数成功返回序列号为index的控制器对象，失败返回null*

Code Example

```
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 ");
    }
}
```

8.2.4 IsLeftHandMode

Method
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

Return

type	desc
bool	*调用函数成功返回值为true：是左手模式，false：不是左手模式*

Code Example

```
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! ");
            }
        }
     }
```

```
}
```

9 IController

interface info
public interface IController
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

9.1 Public Methord Summary

Return type	Method name
bool	*IsAvailable()*
ControllerStatus	*GetControllerStatus()*
int	*ResetCenter()*
ControllerType	*GetControllerType()*
int	*GetBatteryLevel()*
int	*GetPosture(ref Posture pose)*
int	*GetTriggerData(ref float data)*
int	*GetTouchpadTouchPos(ref Vector2 pos)*
int	*GetGyroscope(ref Vector3 gyroscope)*
int	*GetAccelerometer (ref Vector3 accelerometer)*
bool	*IsTouchpadTouching()*
bool	*IsTouchpadTouchDown()*
bool	*IsTouchpadTouchUp()*
bool	*IsTouchpadSwipeUp()*
bool	*IsTouchpadSwipeDown()*
bool	*IsTouchpadSwipeLeft()*
bool	*IsTouchpadSwipeRight()*
bool	*IsButtonPressed(ButtonType button)*
bool	*IsButtonDown(ButtonType button)*
bool	*IsButtonUp(ButtonType button)*
int	*StartVibration()*
int	*StopVibration()*

9.2 Public Methords

9.2.1 IsAvailable

Method
bool IsAvailable()
查询已连接的控制器是否可用

Return

type	desc
bool	*返回true：可用，false：不可用*

Code Example

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

9.2.2 GetControllerStatus

Method
ControllerStatus GetControllerStatus()
获取控制器当前状态

Return

type	desc
ControllerStatus	*返回控制器当前连接状态，控制器状态类型 ControllerStatus 请参考1.3.5 节介绍*

Code Example

```
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;
        }
}
```

9.2.3 ResetCenter（暂不支持）

Method
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）

Return

type	desc
int	*调用函数成功返回0，失败返回-1*

9.2.4 GetControllerType

Method
ControllerType GetControllerType()
获取控制器类型（3Dof、6Dof、游戏、标准输入等）

Return

type	desc
ControllerType	*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1.3.3节介绍*

Code Example

```
if(null == controller){
```

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

9.2.5 GetBatteryLevel

Method
int GetBatteryLevel()
获取控制器电量值

Return

type	desc
int	*调用函数成功返回控制器电量值，失败返回-1*

Code Example

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

9.2.6 GetPosture

Method
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）

Parameters

Parameter name	Parameter desc
pose	*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

Return

type	desc

int	*调用函数成功返回0，失败返回-1*

Code Example

```
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! ");
    }
}
```

9.2.7 GetTriggerData

Method
int GetTriggerData(ref float data)
获取当前控制器的扳机数据

Parameters

Parameter name	Parameter desc
data	*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*

Return

type	desc
int	*调用函数成功返回0，失败返回-1*

TIPS: 当前版本的控制器扳机数据仅为0或1。

Code Example

```
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! ");
    }
}
```

9.2.8 GetTouchpadTouchPos

Method
int GetTouchpadTouchPos(ref Vector2 pos)
获取当前控制器触摸板的触摸位置向量

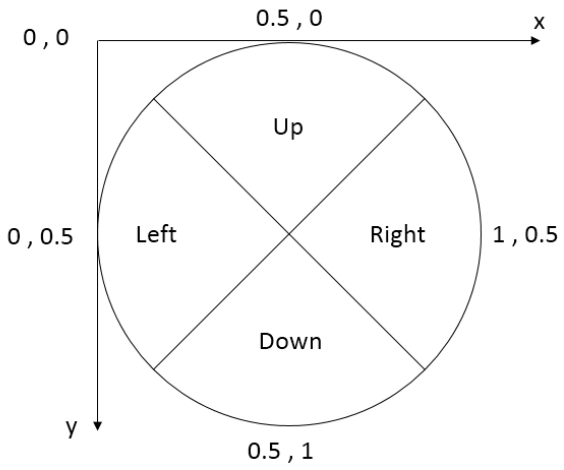
Parameters

Parameter name	Parameter desc
pos	*将当前控制器的位置向量(x,y)写入pos*

Return

type	desc
int	*调用函数成功返回0，失败返回-1*

TIPS: 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:



Code Example

```
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! ");
    }
}
```

9.2.9 GetGyroscope（暂不支持)

Method
int GetGyroscope(ref Vector3 gyroscope)
获取当前控制器的陀螺仪数据

Parameters

Parameter name	Parameter desc
gyroscope	*将当前控制器陀螺仪的位置向量(x,y,z)写入 gyroscope*

Return

type	desc
int	*调用函数成功返回0，失败返回-1*

Code Example

```
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! ");
    }
}
```

9.2.10 GetAccelerometer

Method
int GetAccelerometer(ref Vector3 accelerometer)
获取当前控制器的加速度计数据

Parameters

Parameter name	Parameter desc
accelerometer	*将当前控制器加速度计的位置向量(x,y,z)写入accelerometer*

Return

type	desc
int	*调用函数成功返回0，失败返回-1*

Code Example

```
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! ");
    }
}
```

9.2.11 IsTouchpadTouching

Method
bool IsTouchpadTouching()
判定触摸板当前状态是否被触摸

Return

type	desc
bool	*触摸板状态正在被触摸返回true，否则返回false*

9.2.12 IsTouchpadTouchDown

Method
bool IsTouchpadTouchDown()
判定触摸板是否触发按下动作，兼容HUAWEI VR 1.0的触摸板触摸按下功能

Return

type	desc
bool	*触摸板触发按下动作则返回true，否则返回false*

9.2.13 IsTouchpadTouchUp

Method
bool IsTouchpadTouchUp()
判定触摸板是否触发抬起动作，兼容HUAWEI VR 1.0的触摸板触摸抬起功能

Return

type	desc
bool	*触摸板触发抬起动作则返回true，否则返回false*

9.2.14 IsTouchpadSwipeUp

Method
bool IsTouchpadSwipeUp()
判定触摸板是否触发上滑动作，兼容HUAWEI VR 1.0的触摸板向上滑动功能

Return

type	desc
bool	*触摸板触发上滑动作则返回true，否则返回false*

9.2.15 IsTouchpadSwipeDown

Method
bool IsTouchpadSwipeDown ()
判定触摸板是否触发下滑动作，兼容HUAWEI VR 1.0的触摸板向下滑动功能

Return

type	desc
bool	*触摸板触发下滑动作则返回true，否则返回false*

9.2.16 IsTouchpadSwipeLeft

Method
bool IsTouchpadSwipeLeft ()
判定触摸板是否触发左滑动作，兼容HUAWEI VR 1.0的触摸板向左滑动功能

Return

type	desc
bool	*触摸板触发左滑动作则返回true，否则返回false*

9.2.17 IsTouchpadSwipeRight

Method
bool IsTouchpadSwipeRight ()
判定触摸板是否触发右滑动作，兼容HUAWEI VR 1.0的触摸板向右滑动功能

Return

type	desc
bool	*触摸板触发右滑动作则返回true，否则返回false*

Code Example

```
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 ");
    }
}
```

9.2.18 IsButtonPressed

Method
bool IsButtonPressed(ButtonType button)
判定控制器按键当前状态是否被按下

Parameters

Parameter name	Parameter desc
button	*控制器按键的类型，ButtonType（参考1.3.6 节介绍）*

Return

type	desc
bool	*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*

9.2.19 IsButtonDown

Method
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HUAWEI VR 1.0的返回按键按下功能

Parameters

Parameter name	Parameter desc
button	*控制器按键的类型，ButtonType（参考1.3.6 节介绍）*

Return

type	desc

bool	*对应按键类型的按键触发按下动作则返回true，否则返回false*

TIPS: 当ButtonType为ButtonBack时，在HVR SDK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。

9.2.20 IsButtonUp

Method
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HUAWEI VR 1.0的触摸板单击功能；当ButtonType为ButtonBack时，兼容HUAWEI VR 1.0的返回按键抬起功能

Parameters

Parameter name	Parameter desc
button	*对应按键类型的按键触发按下动作则返回true，否则返回false*

Return

type	desc
bool	*对应按键类型的按键触发抬起动作则返回true，否则返回false*

TIPS:

- 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HVR SDK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）。

Code Example

```
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! ");
    }
}
```

9.2.21 StartVibration（暂不支持）

Method
int StartVibration( )
使控制器开始震动

Return

type	desc
int	*调用函数成功返回0，失败返回-1*

Code Example

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

9.2.22 StopVibration（暂不支持）

Method
int StopVibration()
使控制器停止震动

Return

type	desc
int	调用函数成功返回0，失败返回-1

Code Example

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

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

10 IRenderHandle

interface info
public interface IRenderHandle
提供捕获应用输入的图像，使能色散等显示相关功能

10.1 Public Methord Summary

Return type	Method name
int	*CaptureEyeImage(String filePath)*
void	*EnableChromaticAberration(bool enable)*
int	*GetRenderStatics(ref RenderStatistics renderStatics)*
void	*EnableMonocularDisplay (bool enable)*

10.2 Public Methords

10.2.1 CaptureEyeImage

Method
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）

Parameters

Parameter name	Parameter desc
filePath	*输入图像保存路径*

Return

type	desc
int	*调用函数成功返回0，失败返回-1*

TIPS:

- 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）”

Code Example

```
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! ");
    }
}
```

10.2.2 EnableChromaticAberration

Method
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

Parameters

Parameter name	Parameter desc
enable	*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）*

TIPS:

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。

Code Example

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

10.2.3 GetRenderStatics

Method
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）

Parameters

Parameter name	Parameter desc
renderStatics	*渲染统计信息，RenderStatistics（参考1.4.3节介绍）*

Return

type	desc
int	*调用函数成功返回0，失败返回-1*

Code Example

```
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
}
```

10.2.4 EnableMonocularDisplay

Method
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

Parameters

Parameter name	Parameter desc
enable	*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）*

TIPS:

- 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。

Code Example

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

11 HVREventListener

CLASS info
public class HVREventListener : UnityEngine.EventSystems.EventTrigger, IHVRHoverHandle
提供委托变量，用来传递开发者想要监听的射线指向的UI输入事件（包含：点击、按下、抬起、进入、退出事件）

11.1 Public Delegate Type Summary

Return type	Delegate Type name
delegate void	*VoidDelegate*
delegate void	*EventDelegate*
delegate void	*AxisEventDelegate*

11.1.1 VoidDelegate

Delegate Type
public delegate void VoidDelegate (GameObject go)
该委托类型用于传入发生事件的GameObject

11.1.2 EventDelegate

Delegate Type
public delegate void EventDelegate (GameObject go,PointerEventData eventData)
该委托类型用于传入发生事件的GameObject和PointerEventData

11.1.3 AxisEventDelegate

Delegate Type
public delegate void AxisEventDelegate (GameObject go,AxisEventData eventData
该委托类型用于传入发生事件的GameObject和AxisEventData

11.2 Public Delegate Summary

Delegate type	Delegate name
VoidDelegate	*onEnter*
VoidDelegate	*onExit*
VoidDelegate	*onDown*
VoidDelegate	*onUp*
VoidDelegate	*onClick*
VoidDelegate	*onBeginDrag*
EventDelegate	*onDrag*
EventDelegate	*onDrop*
VoidDelegate	*onEndDrag*
VoidDelegate	*onUpdateSelectObj*
VoidDelegate	*onHover*
AxisEventDelegate	*onMove*

11.3 Public Delegate

11.3.1 onEnter

Delegate
public VoidDelegate onEnter
响应进入事件

11.3.2 onExit

Delegate
public VoidDelegate onExit
响应退出事件

11.3.3 onDown

Delegate
public VoidDelegate onDown
响应进按下事件

11.3.4 onUp

Delegate
public VoidDelegate onUp
响应抬起事件

11.3.5 onClick

Delegate
public VoidDelegate onClick
响应点击事件

Code Example

```
//手柄射线指向退出按钮时，点击触摸板，应用退出（该脚本绑定到退出按钮上）：
void Start () {
    HVREventListener.Get(transform.gameObject).onClick = onPointClick;
}
private void onPointClick(GameObject go){
    if(go == transform.gameObject){
        Application.Quit ();
    }
}
```

11.3.6 onBeginDrag

Delegate
public VoidDelegate onBeginDrag
响应开始进入拖拽事件

11.3.7 onDrag

Delegate
public EventDelegate onEnter
响应拖拽事件

11.3.8 onDrop

Delegate
public EventDelegate onDrop
响应拖拽分离事件

11.3.9 onEndDrag

Delegate
public VoidDelegate onEndDrag
响应结束拖拽事件

11.3.10 onUpdateSelectObj

Delegate
public VoidDelegate onUpdateSelectObj
响应更新获取Obj事件

11.3.11 onHover

Delegate
public VoidDelegate onHover
响应悬停事件

11.3.12 onMove

Delegate
public AxisEventDelegate onMove
响应翻页事件

12 HVRHelpMessage

CLASS info

public class HVRHelpMessage : MonoBehaviour
提供静态变量，用于自定义手柄帮助信息文字、字体颜色、字体大小、背景板颜色等

12.1 Public Static Variable Summary

Variable Type	Variable name
string	*m_trigger_msg*
string	*m_confirm_msg*
string	*m_back_short_msg*
string	*m_back_long_msg*
string	*m_home_short_msg*
string	*m_home_long_msg*
string	*m_volume_msg*
int	*m_FontSize*
Color	*m_ImageColor*
Color	*m_ArrowColor*
Color	*m_TextColor*

12.2 Public Static Variables

12.2.1 m_trigger_msg

Variable
public static string m_trigger_msg
扳机键提示信息

Code Example

```
//自定义扳机键提示信息
void Start () {
    HVRHelpMessage.m_trigger_msg = "trigger msg";
}
```

12.2.2 m_confirm_msg

Variable
public static string m_confirm_msg
触摸板按键提示信息

12.2.3 m_back_short_msg

Variable
public static string m_back_short_msg
返回键短按提示信息

12.2.4 m_back_long_msg

Variable
public static string m_back_long_msg
返回键长按提示信息

12.2.5 m_home_short_msg

Variable
public static string m_home_short_msg
Home 键短按提示信息

12.2.6 m_home_long_msg

Variable
public static string m_home_long_msg
Home 键长按提示信息

12.2.7 m_volume_msg

Variable
public static string m_volume_msg
音量键提示信息

12.2.8 m_FontSize

Variable
public static int m_FontSize
提示信息字号

12.2.9 m_ImageColor

Variable
public static Color m_ImageColor
背景板颜色

12.2.10 m_ArrowColor

Variable
public static Color m_ArrowColor
指示线颜色

12.2.11 m_TextColor

Variable
public static Color m_TextColor
字体颜色

13 HVRArmModel

CLASS info
public class HVRArmModel
提供静态变量，用于自定义3Dof手柄模拟6Dof时手柄的初始位置及肘距

13.1 Public Static Variable Summary

Variable Type	Variable name
Vector3	*m_DefaultControllerPosition*
float	*m_Radius*

13.2 Public Static Variables

13.2.1 m_ DefaultControllerPosition

Variable
public static Vector3 m_DefaultControllerPosition
手柄的初始位置

13.2.2 m_ Radius

Variable
public float Vector3 m_Radius
肘距

14 HVRLinePointer

CLASS info
public class HVRLinePointer : MonoBehaviour
提供与手柄射线光标相关的功能

14.1 Public Methord Summary

Return type	Method name
void	*ShowCircle(bool isTrue)*

14.2 Public Methords

14.2.1 ShowCircle

Method
public void ShowCircle (bool isTrue)
手柄射线指向可交互物体时光标显示圆圈，指向不可交互的物体时光标显示圆点，isTrue为true：显示圆圈，isTrue为false：显示圆点

Parameters

Parameter name	Parameter desc
isTrue	*光标显示形状输入参数（true：显示圆圈，false：显示圆点）*

Code Example

```
//手柄射线指向选择UI时（该脚本绑定到该UI上）
void Start () {
    HVREventListener.Get(transform.gameObject).onEnter = onPointEnter;
}
private void onPoinEnter(GameObject go){
    if(go == transform.gameObject){
        HVRLinePointer.Instance.ShowCircle(true);
```

```
    }
}
```