Coocaa VR 手柄 SDK 使用手册

目录

1 概述

手柄SDK支持Android平台，包括Daydream和Cardboard平台。通过手柄可以获取手柄位置，选取，简化手柄位置、姿态和状态的控制，以及头显的控制。3iVR Unity SDK包括手柄相关的支持、应用和示例程序。通过该使用说明可以方便开发人员创建基于3iVR手柄的应用程序。简化他们的VR开发，比如：

1) 头显回中
2) 手柄追踪
3) 手柄姿态
4) 用户输入事件处理

2 SDK 的使用

2.1 先决条件

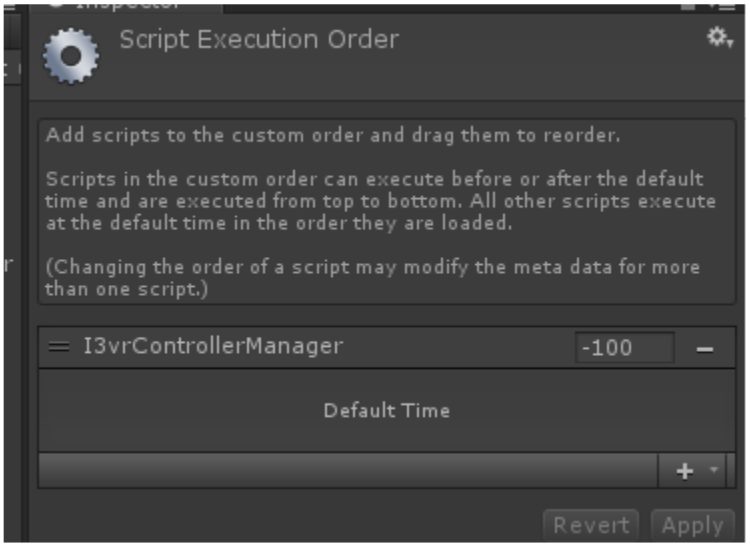
需要Unity 5.4.5及以上版本。在安装Unity时确认已经选取"Android Build Support"组件（默认是已经选取）。如果这是首次在Unity下开发Android应用，请按照官方指南设置Unity的Android SDK（https://docs.unity3d.com/Manual/android-sdksetup.html）。
下载3iVR SDK for Unity（https://github.com/3ivr/3ivr-unity-sdk && https://github.com/CoocaaVR/CCVRSDK）。包括SDK和demo程序。
当然你还需要 Coocaa 一体机和 3i 手柄。
保持 VR 设备的蓝牙为打开状态。

2.2 导入 SDK

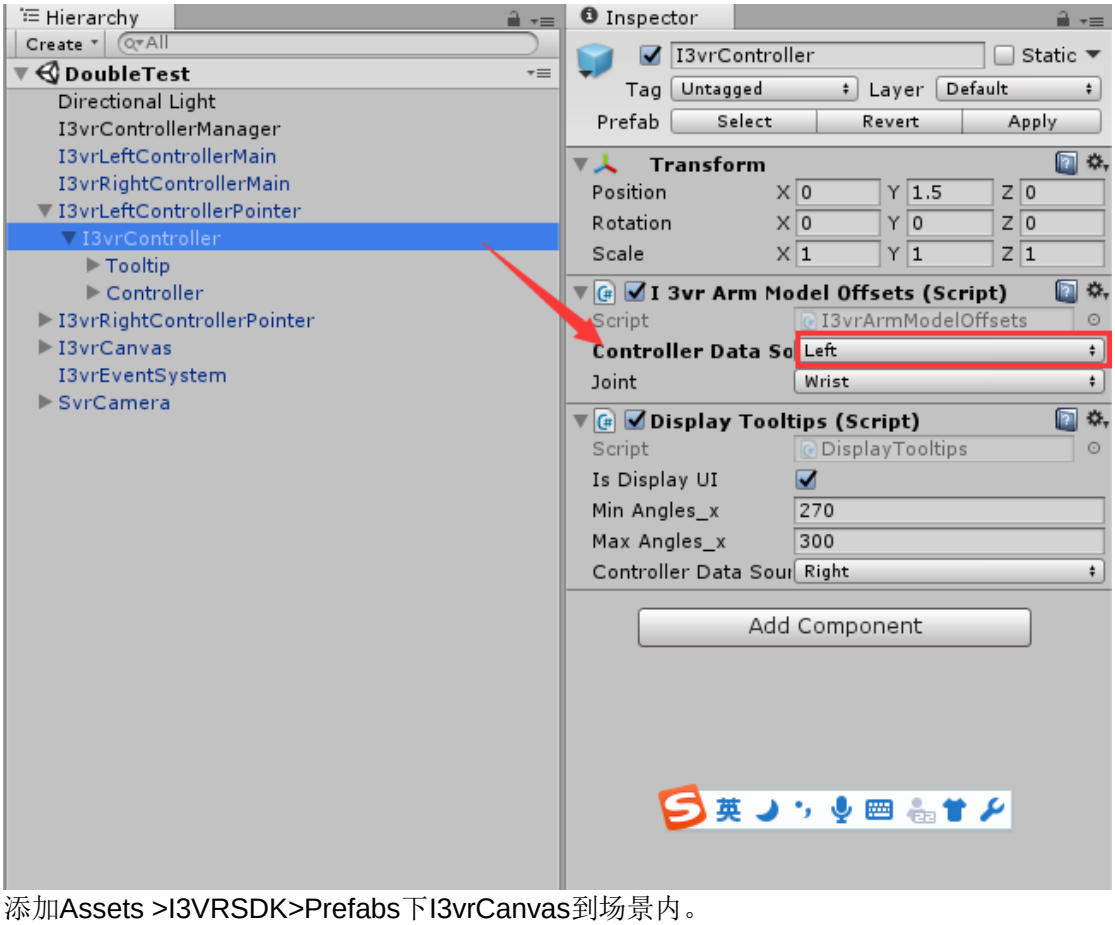
1） 打开 Unity 5.4.5 或者以上版本，创建或打开一个工程项目
2） 找到选项Assets > Import Package > Custom Package。打开并 选中3ivrSdkForUnity文件，检查导入资源包内所有资源是否选中，然后单击Import。如果出现提示，请接受升级。
3） 同 2），选中 CoocaaVR_SDK 文件，检查导入资源包内所有资源是否选中，然后单击 import。

2.3 启用 I3vr 控制器和 UI 的支持

1） 创建一个空物体，并命名为 I3vrControllerManger。
2） 在Project面板内搜索I3vrControllerManager类，在Inspector面板点选Execution Order拖入该类



3） 挂载I3vrControllerManager类，并设置Controller Type，双手柄为Left And Right。
4） 添加Assets >I3VRSDK>Prefabs下I3vrRightControllerMain到场景内(双手柄模式下还需添加I3vrLeftControllerMain)。
5） 添加Assets >I3VRSDK>Prefabs下I3vrRightControllerPointer到场景内(双手柄模式下还需添加I3vrLeftControllerPointer)。
6） 根据需求设置手柄数据来源，分别设置I3vrLeftControllerPointer 下I3vrController的ControllerDataSource为Left，I3vrRightControllerPointer下I3vrController的ControllerDataSource为Right。



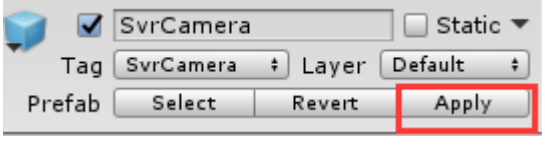
7） 添加Assets >I3VRSDK>Prefabs下I3vrCanvas到场景内。

8） 添加Assets >I3VRSDK>Prefabs下I3vrEventSystem到场景内。

9） 将 Assets > SVR >Prefabs 下 SvrCamera 拖到场景内。

10） 将 MainCamara 拖到 SvrCamera 下 Head。将 MainCamera 的 CulingMask 设置 Nothing。

11）MainCamera 的 Tag 值为 MainCamera，并确保 SvrCamera、I3vrLeftControllerPointer、I3vrRightControllerPointer 的 Transform 组件 Position 和 MainCamera Position 数值一致。（注意： Apply）



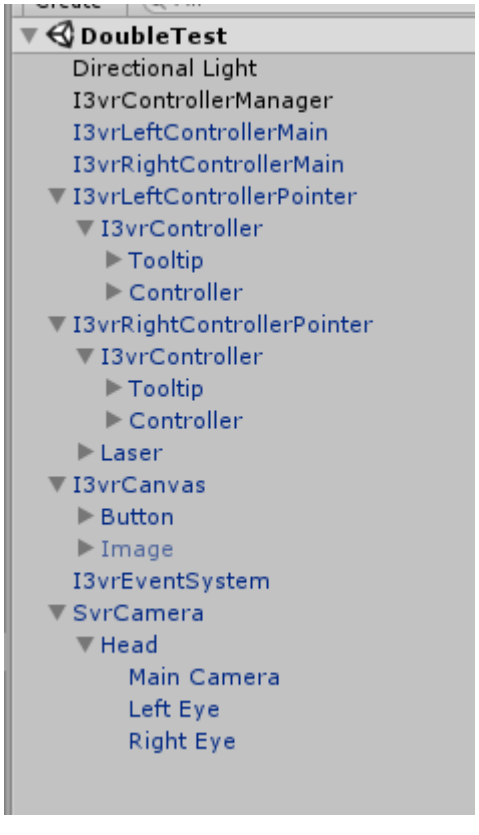
12） 打开 SvrCamera 挂载的脚本 SvrManager，注释红色圈内代码

```
    private IEnumerator Initialize()
    {
        // Plugin must be initialized OnStart in order to properly
        // get a valid surface
        GameObject mainCameraGo = GameObject.FindWithTag("MainCamera");
        if (mainCameraGo)
        {
//          mainCameraGo.SetActive(false);

            Debug.Log("Camera with MainCamera tag found.");
            if (!disableInput)
            {
                Debug.Log("Will use translation and orientation from the MainCamera.");
                transform.position = mainCameraGo.transform.position;
                transform.rotation = mainCameraGo.transform.rotation;
            }

            Debug.Log("Disabling Camera with MainCamera tag");
        }
```

13） 保存场景。



2.4 示例场景

Demo：DoubleMain.apk
Demo：Coocaa3ivrsdkTestBtn.apk
使用方法：安装Demo到vr设备，保持vr设备蓝牙开启状态，打开应用模块内安装的demo，长按手柄主页键开机自动连接即可使用手柄。

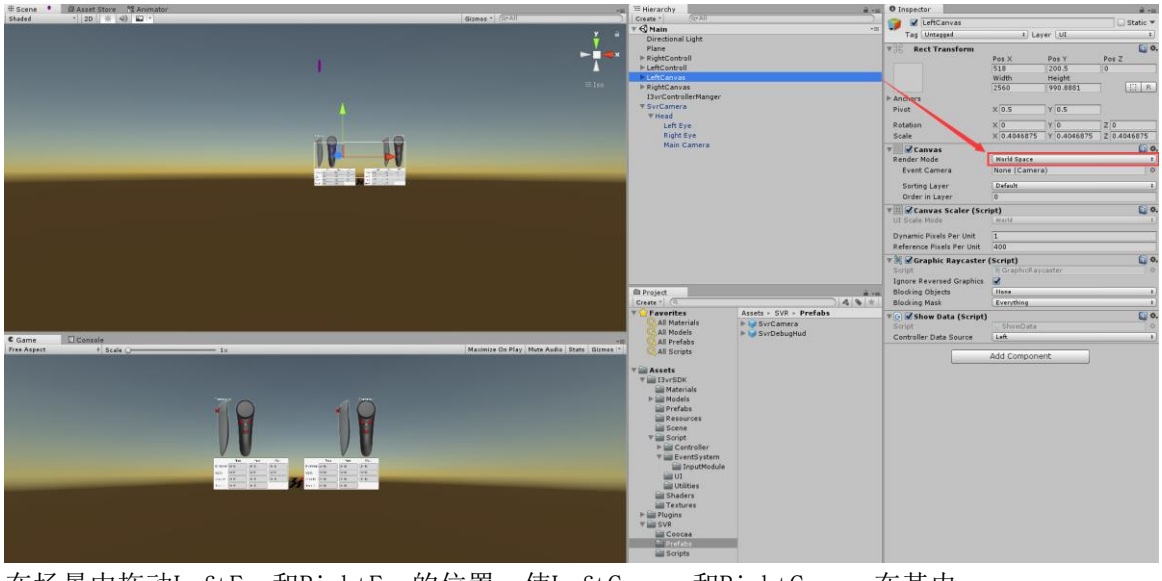
1) Assets>I3VRSDK>Scene>Main.unity
   a) 按钮状态显示，触摸点触点位置，触摸板手势触发。

b) 按钮检测到ButtonDown切换绿色，检测到ButtonUp切换红色。
c) 触摸点检测到TouchDown，显示绿点，并显示对应的触点位置。检测到TouchUp时隐藏。
d) 触摸手势触发会显示对应手势箭头(左划，右划，上划，下划，在显示0.1S后隐藏)。
e) 手柄实时姿态展示。
f) Rotation，Gyro，Accele，Touch Pos对应API读数显示。

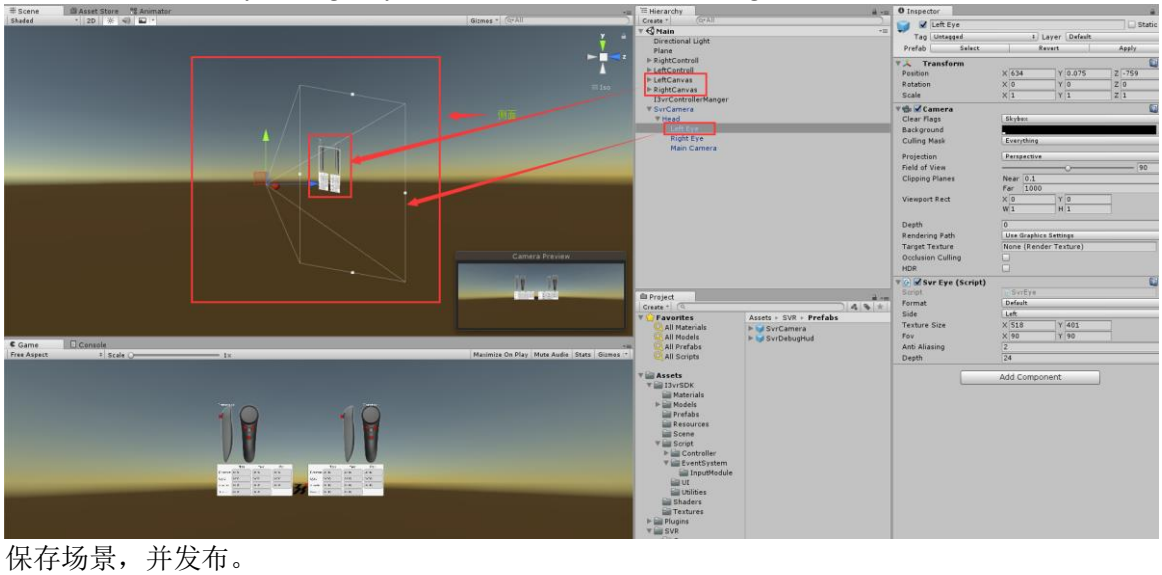
注意：若是用Main.Unity场景，设置步骤：

（1） 需要拖入SVRCamera，将MainCamera拖入SVRCamera>Head下，并将MainCamera的cullingMask设置为Nothing。

（2） 设置LeftEye和RightEye的Inspector内的RenderMode为WorldSpace。



（3） 在场景中拖动LeftEye和RightEye的位置，使LeftCanvas和RightCanvas在其内。



（4） 保存场景，并发布。

2) Assets>I3VRSDK>Scene>TestButton.unity
a) UGUI-Button触发展示。
b) Arm Model 手臂模型展示。
c) 激光光标与按钮提醒展示。

2.5 发布设置

1) 找到选项File>Build Settings，将场景添加到场景列表，并确认为首个启动的场景，将发布平台选择为Android平台，并点击Switch PImport，确认Unity图标在对应Android平台。
2) 选中Player settings，找到Other Settings并执行以下操作。
3) Bundle Identifier 输入包名(例如com.coocaa.UnityDemo)。
4) Minimum API Level 设置为Android 5.0“Lollipop”(API LEVEL 21)或以上。

2.6 3iVR 控制器支持

1) Arm Model：使得VR中的控制器模型接近于I3vr控制器物理位置的数学模型。
2) Input System：控制器模型射线交互模块,包含输入模块和接收模块,使得更好的通过手柄与UI或其他对象进行交互。
3) Controller visualization：I3vr控制器的3D模型,按钮提醒,触摸板触点可视化。
4) Laser and reticle visualization：显示激光笔和光标,试用户轻松的和VR环境交互。

2.6.1 Arm Model

提供了一个手臂模型，一个使用物理控制器的方向和位置的数学模型，并且预测肩部，肘部，手腕和指针的位置，以确定在场景中放置控制器模型的位置。手臂模型通过I3vrArmModel脚本进行控制，该脚本附加到提供的预制I3vrControllerMain。
❖ 注意：场景中必须只有一个I3vrArmModel实例。

该I3vrArmModel脚本允许您调整以下属性：
1) 添加肘部高度：增加了一个偏移量来预测肘部的高度。
2) 添加肘部深度：增加一个偏移量预测肘部深度。
3) 指针倾斜角度：相对于控制器的指针向下倾斜的角度。对于和眼睛水平的物体进行交互时，15°的默认值是比较舒适的。如果用户需要与显着高于或低于对象的对象进行交互，则可能需要调整此属性。
4) 脸部淡出距离：控制器与脸部之间的距离，此后α值降低。
5) 追随目光：确定肩膀是否应该跟随用户的目光。
6) 使用加速度计：关闭加速器使控制器感觉更准确和可预测。打开它减少精度，但可以使用户感觉更像是持有一个真正的对象。
❖ 注意：默认设置被调整，假设用户将像激光笔一样握住控制器。如果您的应用程序需要使用控制器进行诸如摆动蝙蝠或捕捉球的操作，则需要适当地调整设置。

2.6.2 Input System

要想使用此系统，请添加预制体I3vrEventSystem到场景中，你可以通过I3vrPointerInputModule脚本来配置输入系统。

2.6.3 Laser and reticle visualization

预制体I3vrControllerPointer下Laser挂载I3vrLaserPointer脚本控制着激光可视化和显示光标。该脚本具有以下设置：

1) 激光颜色：确定激光的颜色。
2) 最大激光距离：当您目前没有指向物体时，激光显示的最大距离（以米为单位）。
3) 最大刻线距离：当您目前未指向对象时，这是激光光标显示的最大距离（以米为单位）。

2.6.4 Controller Visualization

预制体I3vrControllerPointer下I3vrController对象下挂载的DisplayTooltips脚本控制触点可视化和按钮提醒UI,有如下设置：

1) 是否显示提醒UI。
2) X轴最小显示角度：大于该值小于最大角度时显示UI，为顺时针旋转值。
3) X轴最大显示角度：小于该值大于最小角度时显示UI该值为顺时针旋转值。

3 API 参考

手柄（控制器）按钮以及触摸板和各数据接口封装为 I3vrController.cs。
设置手柄类型，获取左右手实例：I3vrControllerManager.cs
激光射线点:I3veLasterPointer.cs
根据手臂模型的关节来确定位置和旋转变换：I3vrArmModelOffsets.cs
控制器和场景之间的标准接口:I3veArmModel.cs
接口用法：参照 showData.cs
详情如下：

3.1 I3vrController

Summary:

3iVR controller API的主入口。
为了使用API，增加这个对象到你场景中的GameObject中，或者使用I3vrControllerMain预制体。
为了存取控制器的状态，简单读取该类的属性。

Inheritance:

```
Inherits from: MonoBehaviour
```

Events:

OnControllerUpdate	OnControllerUpdateEvent

HeadsetRecenter	OnHeadsetRecenter

Properties:

ArmModel	I3vrArmModel 返回手臂与控制器关联模型实例。
ConnectionState	I3vrConnectionState 返回当前控制器连接状态。
ApiStatus	I3vrControllerApiStatus 返回当前控制器API状态。
Error	连接错误；
Disconnected	断开连接；
Scanning	搜索手柄；
Connecting	正在连接手柄；
Connected	已连接手柄
Orientation	Quaternion 返回控制器当前的空间方向，作为四元数。表示方向的空间是通常的 Unity 空间， X 指向右边，Y 向上，Z 指向前方。 所以在场景中要创建一个具有与控制器相同的方向的对象，只需分配一下四元数到 GameObject 的 transform.rotation。
Gyro	Vector3 返回控制器的陀螺仪读数。 陀螺仪表示其每个局部轴的角度。 控制器的轴是：X 指向右边，Y 从控制器的顶面垂直向上指向，Z 位于控制器的正面，指向前方。 角速度以弧度/秒给出，使用右手规则（正数表示围绕给定轴的右旋转）。
Accel	Vector3 返回控制器的加速度计读数。 加速度计表示加速度和重力在每个控制器局部轴的方向上的影响。控制器的局部轴为：X 指向右侧，Y 从控制器顶部表面垂直向上，Z 位于控制器的正面，指向前方。 加速度以米/秒为单位进行测量。 请注意，重力与加速度相结合，因此当控制器靠在桌面上时，它将测量 Y 轴上的 9.8 m / s ^ 2 的加速度。 只有控制器处于自由落体状态，或者用户处于零重力环境（如空间站）时，所有三个轴上的加速度计读数将为零。
IsTouching	bool 如果为 **true**，则用户正在触摸控制器的触摸板。
TouchDown	bool 如果为 true，用户刚刚开始触摸触摸板。 这是一个事件标志：事件发生后只有一帧是 true，然后恢复为 false。

TouchUp	bool 如果为 true，用户刚刚停止触摸触摸板。 这是一个事件标志：事件发生后只有一帧是 true，然后恢复为 false。
TouchPos	Vector2 返回触摸板触点读数。原点位于矩形左上角 (0，0)，右下角为(1，1)。
Recentered	bool 如果为 true，用户刚刚完成了手柄回中。控制器的方向现在在新的坐标系中被报告（控制器的方向在重新设计完成时被重新映射为“前进”）。 这是一个事件标志（事件发生后只有一帧是 true，然后恢复为 false）。
TriggerButton	bool 扳机按钮. 如果为真，则触发按钮（手柄按钮）当前正在按下。 这不是一个事件：它代表按钮的状态（按住按钮时它保持为真）。
TriggerButtonDown	bool 如果为 true，则按下触发按钮。 这是一个事件标志：事件发生后只有一帧是 true。
TriggerButtonUp	bool 如果为 true，则触发按钮刚刚释放。 这是一个事件标志：事件发生后只有一帧是 true。
AppButton	bool 如果为 true，则当前正在按下应用按钮（手柄按钮）。这不是一个事件：它代表按钮的状态（按住按钮时它保持为 true）。
AppButtonDown	bool 如果为 true，应用按钮刚刚被按下。 这是一个事件标志：事件发生后只有一帧是 true。
AppButtonUp	bool 如果为 true，则应用按钮刚刚被释放。 这是一个事件标志：事件发生后只有一帧是 true。
HomeButton	bool 如果为 true，则按住主按钮（手柄按钮）。 这不是一个事件：它代表按钮的状态（按下按钮时它保持为 true）。
HomeButtonDown	bool 如果为 true，则按住主屏幕按钮。 这是一个事件标志：事件发生后只有一帧是 true。
HomeButtonUp	bool 如果为 true，则主页按钮刚刚被释放。

	这是一个事件标志：事件发生后只有一帧是 true。
SwitchButton	bool 如果为 true，则切换按钮（手柄按钮）当前正在按下。 这不是一个事件：它代表按钮的状态（按住按钮时它保持为 true）。
SwitchButtonDown	bool 如果为 true，则按下开关按钮。 这是一个事件标志：事件发生后只有一帧是 true。
SwitchButtonUp	bool 如果为 true，则开关按钮刚刚被释放。 这是一个事件标志：事件发生后只有一帧是 true。
TouchGestureLeft	bool 如果是true，则用户触发了左划手势。 这是一个事件标志：事件发生后只有一帧是 true。
TouchGestureRight	bool 如果是true，则用户触发了右划手势。 这是一个事件标志：事件发生后只有一帧是 true。
TouchGestureUp	bool 如果是true，则用户触发了上划手势。 这是一个事件标志：事件发生后只有一帧是 true。
TouchGestureDown	bool 如果是true，则用户触发了下划手势。 这是一个事件标志：事件发生后只有一帧是 true。
ErrorDetails	string 如果State == I3vrConnectionState.Error，则包含有关错误的详细信息。

3.2 I3vrArmModel

Summary:

I3vrArmModel是控制器和场景之间的标准接口，其主要负责：

- 检测手柄的方向和位置
- 预测肩膀、肘部、手腕和指针的位置

Inheritance:

Inherits from: MonoBehaviour

Public types:

GazeBehavior	Enmu 设置追随模式

Properties:

Instance	I3vrArmModel 使用I3vrController的对象获取此类的单例。
pointerPosition	Vector3 表示指针位置的向量。
pointerRotation	Quaternion 表示指针旋转的四元数。
wristPosition	Vector3 表示手腕位置的向量。
wristRotation	Quaternion 表示手腕旋转的四元数
elbowPosition	Vector3 表示肘部位置的向量。
elbowRotation	Quaternion 表示肘部旋转的四元数
shoulderPosition	Vector3 表示肩膀位置的向量。
shoulderRotation	Quaternion 表示肩膀旋转的四元数。
alphaValue	Float 控制器射线的alpha值。

Public attributes:

addedElbowHeight = 0.0f	Float 肘部高度（m）。
addedElbowDepth = 0.0f	Float 肘部深度（m）。
pointerTiltAngle = 15.0f	Float 激光指示器相对于控制器的向下倾斜的角度。
fadeDistanceFromFace = 0.32f	Float 控制器距离摄像机消失距离(m)。
followGaze = GazeBehavior.DuringMotion	GazeBehavior 确定肩膀是否应该追随目光。
useAccelerometer = false	Bool 是否开启加速度计。

3.3 I3vrArmModelOffsets

Summary

该脚本根据手臂模型的关节来确定位置和旋转变换。

Inheritance

Inherits from: MonoBehaviour

Public types

Joint	Enum 关节设置

Public attributes

joint	Joint 确定哪个关节设置位置和旋转。
ControllerDataSource	DataSource 设置手柄数据来源

3.4 I3vrLaserPointer

Summary

激光射线点附属于Controller对象。激光射线可以帮助用户定位光标。

Inheritance

Inherits from: MonoBehaviour

Public attributes

laserColor = new Color(1.0f, 1.0f, 1.0f, 0.25f)	Color 激光指示器的颜色包括alpha透明度。
maxLaserDistance = 0.75f	Float 指针的最大距离（m)。
maxReticleDistance = 2.5f	Float 光标的最大距离（m)。
reticle	GameObject 光标对象。

3.5 I3vrControllerManager

Summary

设置手柄类型，获取左右手实例。

Inheritance

Inherits from: MonoBehaviour

Public types

ControllerType	enum 设置手柄类型。
DataSource	enum 数据来源

Public Properties

I3vrRightController	I3vrController 右手手柄对象
I3vrLeftController	I3vrController 左手手柄对象
I3vrControllerNumb	ControllerType 手柄类型