**Holoever XR SDK**

Unity 开发指南

### 版本：V2.2.1



# 版本更新说明

|  |  |
| --- | --- |
| 版本号 | 更新内容 |
| 2.2.0 | 新增slam 插件 |
| 2.1.0 | 1. 补全部分英文注释，提供API   Renference 文档。   1. 实现并提供一套开发者可以自定义手柄模型、流程的demo。 2. 优化关机对话框 UI 显示。 3. 优化手柄校准提示框判断逻辑。 4. 新增手柄按键提示 UI。 5. 新增NEO2 手柄键值（A/B/X/Y） 支持。 6. 实现对Unity 的 Multithreaded Rendering 的支持。 7. 部分配置改用SettingsAssetConfig.asset 机制，持久化； 8. 修复已知bug |
| 2.0.1 | 1. 支持Universal RP。 2. 修复已知BUG。 |
| 2.0.0 | 1. 新增手柄模型动态加载功能。 2. 新增签名验证机制。 3. 支持arm64 库。 |

|  |  |
| --- | --- |
|  | 4.性能优化。 |
| 1.0.4 | 1. 支持Unity5.4.0f3 或更高版本。 2. 修复已知BUG。 |
| 1.0.2 | 1. 优化键值接口，调用更方便。 2. 新增权限申请接口，和物体拖拽示例。 3. 整理优化文档结构。 4. 修复已知BUG。 |
| 1.0.1 | 1. 优化手柄相关接口和脚本性能优化。 2. 升级到Unity2018.4.0 LTS 版本。 3. 修复已知BUG。 |
| 1.0.0 | 发布 1.0.0 版本 |

目录

[版本更新说明 3](#_bookmark0)

1. [说明 7](#_bookmark1)
   1. [SDK 介绍 7](#_bookmark2)
   2. [文档介绍 8](#_bookmark3)
2. [支持设备 8](#_bookmark4)
3. [开发环境要求 8](#_bookmark5)
   1. [Android 开发环境配置 8](#_bookmark6)
   2. [URP 开发环境配置 8](#_bookmark7)
4. [开发注意事项 9](#_bookmark8)
   1. [Android 手机模拟运行 9](#_bookmark9)
   2. [Unity 注意事项 9](#_bookmark10)
   3. [Android Manifest 文件 11](#_bookmark11)
   4. [SDK 推荐配置 12](#_bookmark12)
   5. [Universal RP(通用渲染管线)配置说明 14](#_bookmark13)
5. [SDK 接入快速入门 16](#_bookmark14)
   1. [新建工程，导入插件 16](#_bookmark15)
   2. [预制体的使用 16](#_bookmark16)
   3. [模拟运行 16](#_bookmark17)

[5.4 打包 17](#_bookmark18)

1. [按键手柄交互 18](#_bookmark19)
   1. [一体机按键 18](#_bookmark20)
   2. [手柄按键 19](#_bookmark21)
2. [API 接口函数一览 22](#_bookmark22)
   1. [相机FOV 参数调节 22](#_bookmark23)
   2. [一体机性能等级参数 22](#_bookmark24)
   3. [软件盘输入控件 22](#_bookmark25)
   4. [获取头部姿态数据 23](#_bookmark26)
   5. [相机预览画面 23](#_bookmark27)
   6. [按键功能接口 29](#_bookmark28)
   7. [视角锁定和重置接口 29](#_bookmark29)
   8. [纹理质量 30](#_bookmark30)
   9. [6DOF 手柄支持 30](#_bookmark31)
   10. [系统相关接口 31](#_bookmark32)
   11. [Pro 版本高级服务 33](#_bookmark33)
       1. [授权相关说明 33](#_bookmark34)
       2. [插件验证结果 34](#_bookmark35)
       3. [插件支持状态 34](#_bookmark36)
       4. [6DOF 位移 35](#_bookmark37)
       5. [视频录制 36](#_bookmark38)
       6. [Marker 识别 38](#_bookmark39)
   12. [物体拖拽 40](#_bookmark40)
   13. [权限申请 41](#_bookmark41)
   14. [APK 加密功能 42](#_bookmark42)
   15. 多线程渲染 43
3. [辅助工具 44](#_bookmark43)
4. [常见问题解答 45](#_bookmark44)

### 说明

* 1. SDK 介绍

Holoever MR SDK 是一整套针对 Holoever VR/AR OS 开发的工具包，工具包面向VR/AR 设备提供了 VR/AR 数据接口，让开发者能够快速开发 VR/AR 设备的应用和游戏。XR SDK 包含的功能有：

1. 左右分屏画面，针对左右眼分别渲染左右分屏画面，保持 3D 景深；
2. 头部随动，低延迟随动支持，根据一体机调优处理；
3. 反畸变，根据透镜参数进行反畸变反色散处理，在一体机中显示不失真；
4. VR/AR 显示参数，SDK 在一体机运行时动态加载显示参数，应用无需处理不同设备的不同参数就可获得一致的显示效果；
5. 辅助功能，SDK 将逐步完善各种辅助功能，如 Holoever 键重置头部位置，物体瞄准，按键监听处理，帧率统计等。
6. 独有的DTR 动态时间帧补偿技术，让画面保持满帧且不卡顿。

* 1. 文档介绍

本文档主要介绍了 SDK 的接入方法，包括：立体渲染、传感器融合以及设备控制器的使用等功能的接入。

并介绍了 SDK 的目录结构与接口 API 的功能。

另外，还对一些常见疑问进行了解答。开发者使用 SDK 前请务必阅读该文档对应的内容，以提升开发效率。

### 支持设备

SDK 支持运行Holoever OS 系统的一体机，包括VR/AR 设备。

### 开发环境要求

SDK 推荐使用Unity2018.4.0f1，也支持 Unity2018.4.5.f1/Unity2019.3 等版本。

不推荐使用Unity2019.4.x LTS 版本，经测试发现存在内存泄漏问题。

建议开发者选用LTS 版本，更稳定些，由于不同的Unity 版本有些差异比较大，如果使用的非推荐版本遇到问题后，可以先尝试换成推荐版本。

## Android 开发环境配置

|  |  |
| --- | --- |
| 软件名称 | 软件版本 |
| JDK | JDK 1.8.0 及以上 |
| Android SDK | API Level 19 及以上 |

* 1. URP 开发环境配置

|  |  |
| --- | --- |
| 软件名称 | 软件版本 |
| Unity | 2019.3.6 |
| URP | 7.1.8 |

### 开发注意事项

* 1. Android 手机模拟运行

没有一体机的情况下，直接在Android 手机上运行将只能单屏显示，如果要进行调试开发，请使用HoloeverMR启动器打开应用。 为了让启动器能够正常识别开发的应用，需要检查 AndroidManifest.xml 中是否声明了 category

(<category android:name="com.nibiru.intent.category.NVR" />) 。

如果activity 是AR/VR 类型的，标记NVR 标签com.nibiru.intent.category.NVR

如果是普通 2D 界面，标记 2D 标签 com.nibiru.intent.category.2D 如果是系统渲染界面，如果是TV 模式，标记 TV 系统渲染标签

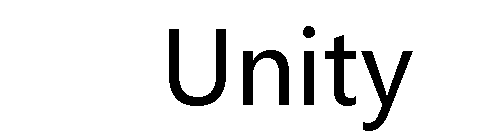
com.nibiru.intent.category. SYS\_RENDER\_TV，如果是 PHONE 模式，标记

PHONE 系统渲染标签 com.nibiru.intent.category.SYS\_RENDER\_PHONE

我们强烈建议使用Nibiru VR/AR 设备进行开发和调试。

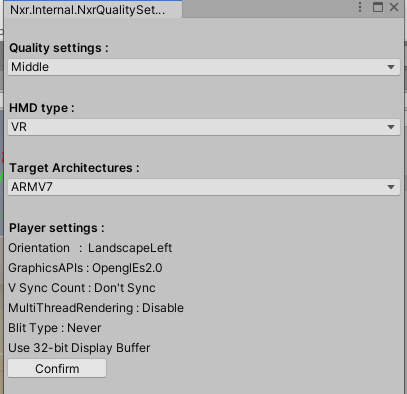
## Unity 注意事项

使用前请注意以下事项：

版本不同 FPS 帧率可能存在差别。

在Unity 编辑器运行模式下，使用 Alt+鼠标移动相当于头部转动效果， 使用Ctrl+鼠标移动相当于头部绕 z 轴转动。

为了方便开发者，SDK 在编辑器提供了一键XR 配置界面，位于工具栏HoloeverXR - XR Settings，界面如下：



Quality settings 包含Low/Middle/High Low : 关闭抗锯齿

Middle : 2 倍抗锯齿High：4 倍抗锯齿

HMD Type 当前开发的目标类型，必须选择

VR : 针对VR 机器开发的应用 AR : 针对 AR 机器开发的应用

Target Architectures 打包 32 位还是 64 位，默认 32 位

64 位需要系统支持的情况下，才可以正常运行。Player settings 修 改 默 认 打 包 配 置 参 数 ： Orientation : LandscapeLeft

GraphicsAPIs : OpenglEs2.0 Use 32-bit Display Buffer

点击Confirm 后就会自动修改配置。

## Android Manifest 文件

如果应用中不包含AndroidManifest 文件，请直接使用 SDK 中的AndroidManifest 文件。如果应用中包含，需要开发者合并AndroidManifest 文件。

* Activity 需要继承自

### com.nibiru.lib.xr.unity.NibiruXRUnityActivity.

Intent-filter 需要添加：

<category android:name="com.google.intent.category.CARDBOARD" />

<category android:name="com.nibiru.intent.category.NVR" />

<category android:name="com.nibiru.intent.category.STUDIO" />

* 插件声明：

<!-- "6DOF", "RECORD", "MARKER" !-->

<meta-data android:value="6DOF " android:name="NIBIRU\_PLUGIN\_IDS"/>

<meta-data android:value="NxrViewerMain" android:name="NIBIRU\_UNITY\_VIEWER\_NAME"/>

* 当前开发的目标设备类型：

<!--0=vr,1=ar-->

<meta-data android:value="VR" android:name="HMD\_TYPE"/>

* 当前是否为加密版本：

<!-- 当前APK为加密版本-->

<meta-data android:value="0" android:name="NIBIRU\_ENCRYPTION\_MODE"/>

* 添加必要的权限：

<uses-permission android:name="android.permission.BLUETOOTH\_ADMIN"/>

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

<uses-permission android:name="android.permission.ACCESS\_COARSE\_LOCATION" />

<uses-permission android:name="android.permission.WRITE\_EXTERNAL\_STORAGE"/>

<uses-permission android:name="android.permission.READ\_EXTERNAL\_STORAGE" />

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

<uses-permission android:name="android.permission.GET\_TASKS" />

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

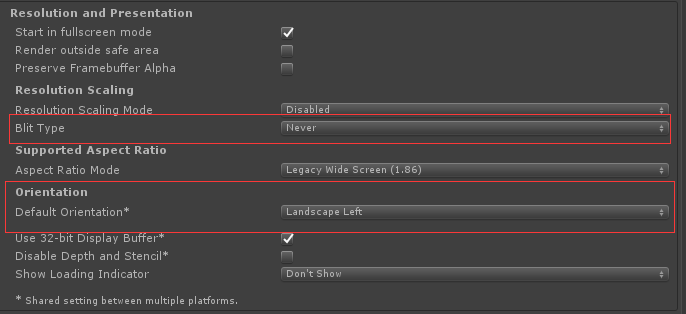
<uses-permission android:name="android.permission.MOUNT\_UNMOUNT\_FILESYSTEMS"/>

<uses-permission android:name="android.permission.ACCESS\_NETWORK\_STATE"/>

<uses-permission android:name="android.permission.WRITE\_SETTINGS"/>

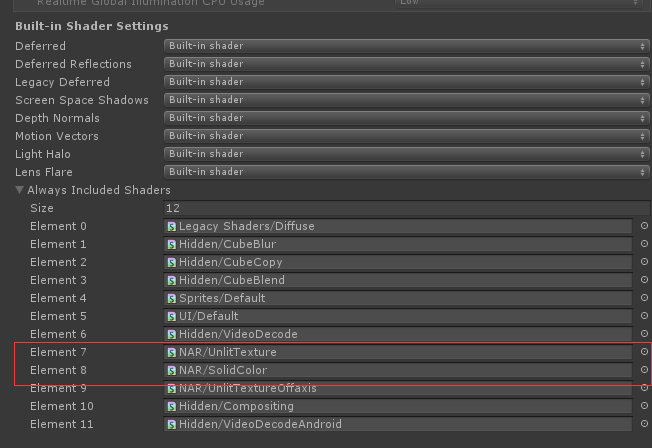
<uses-permission android:name="android.permission.ACCESS\_WIFI\_STATE"/>

## SDK 推荐配置

* Graphics APIS 不支持Vulkan，对于OpenGLES2,OpenGLES3，开发者根据自身需求自行选择。
* MultiThreaded Rendering 暂不支持，不需要勾选。
* Unity Player Setting 中Default Orientation 需要选择 Landscape Left Resolution Scaling 下面的Blit Type 需要选择为Never
* Quality 中的V Sync Count 需要选择为 Don’t Vsync



* Edit/Project Settings/Graphics，将Resource/UnlitTexture.shader 和Resource/SolidColor.shader 加入到Always Included Shaders 列表。

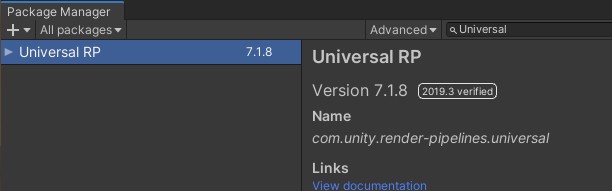


## Universal RP(通用渲染管线)配置说明

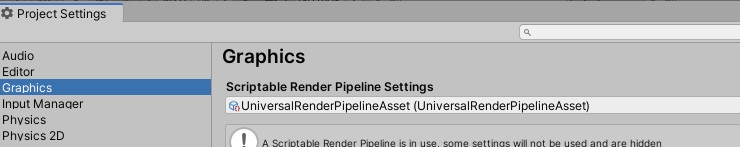
Unity 版本：Unity2019.3.6 URP 版本：V7.1.8

如果项目工程使用了URP 请参考当前配置说明进行配置适配。

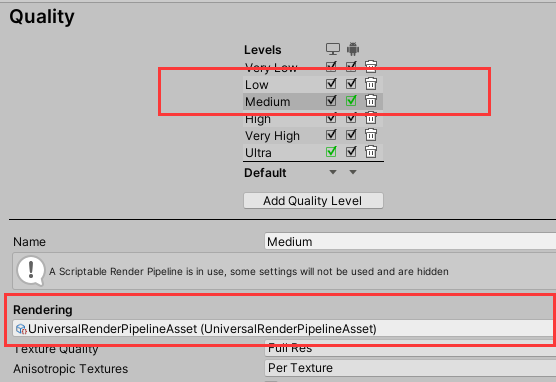
* + 打开Window/Package Manager，搜索Universal RP，点击Install。



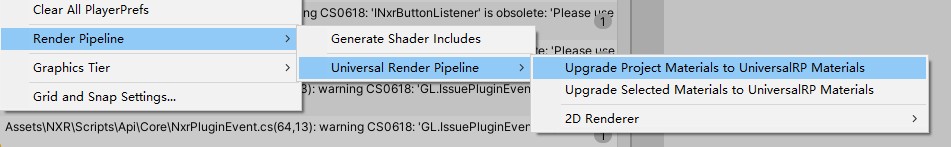
* + 安 装 完 成 ， 创 建 UniversalRenderPipelineAsset， Assets/Create/Remdering/Universal Render Pipeline/Pipeline Asset
  + 打开Edit/Project Settings/Graphics/，在 Scriptable Render Pipeline Settings 中选择已创建的PipelineAsset。



* + 打开Edit/Project Settings/Quality 界面，选择Android 类别，在Rendering 中选择已创建的PipelineAsset。



* + 升级工程材质



注：取消UniversalRenderPipelineAsset 中的SRP Batcher/Dynamic Batching



否则会出现内存泄露问题。

### SDK 接入快速入门

新建工程，导入插件

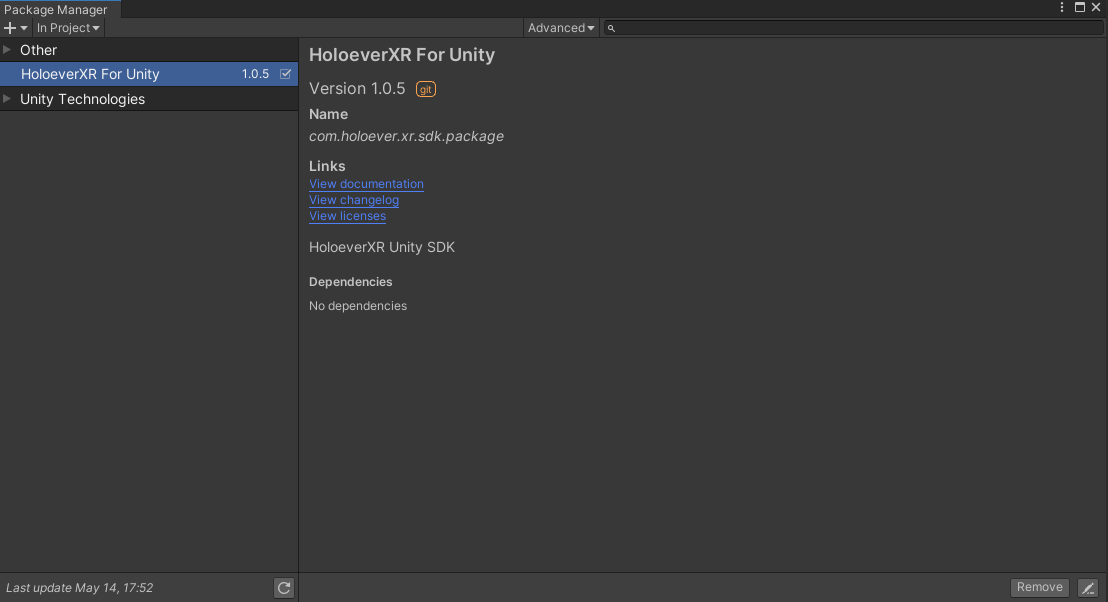
点击 Windows/PackageManager,打开



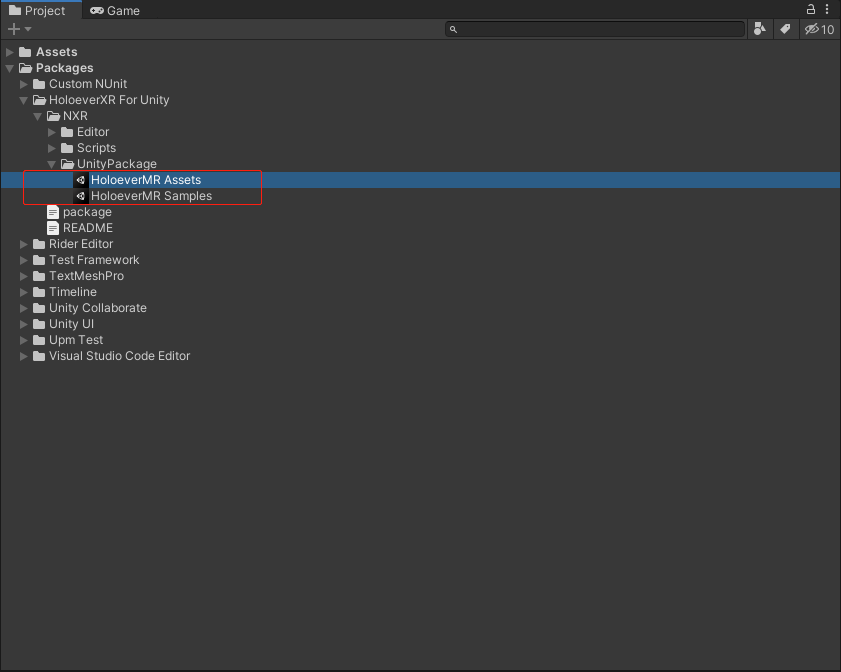
点击左上角+ 选择 [add package from git URL]

填入地址"https://github.com/Kong-W/HoloeverMR.git" 点击[add]

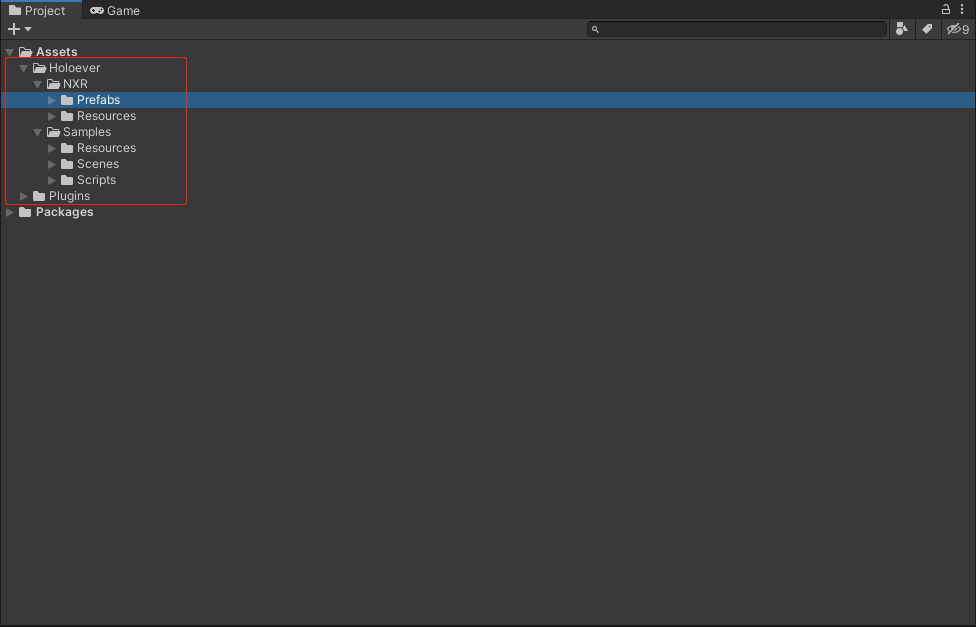
点击左下角刷新按钮



等待加载完成



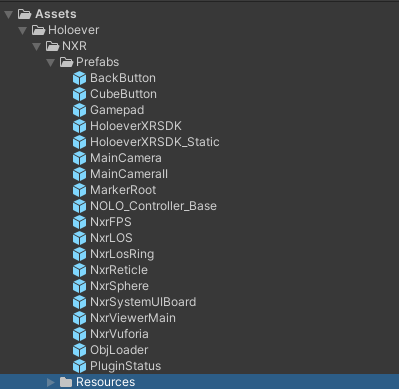
找到如上图位置 双击导入示例



SDK导入完成

### 预制体的使用

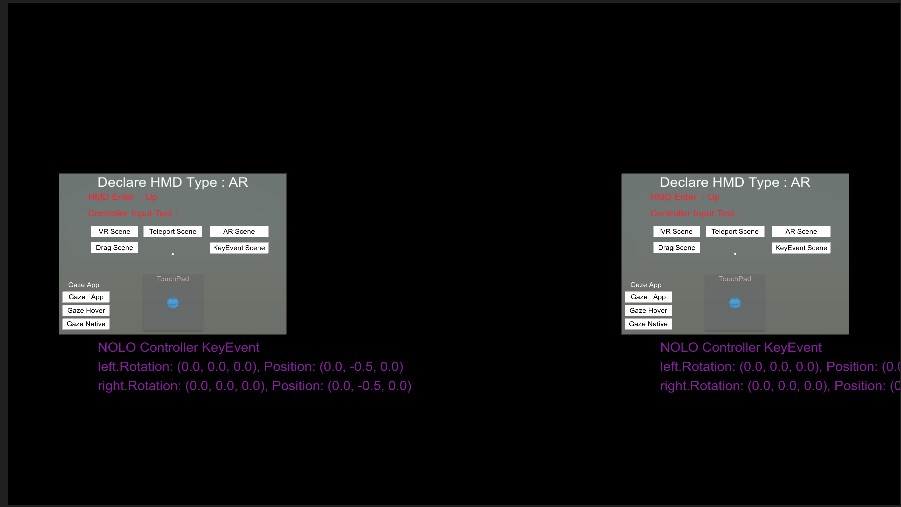
在NXR/Prefabs 目录下，包含了 SDK 常用的预制体组件。



将HoloeverXRSDK 预制体拖放到场景内，将Transform 组件的Position 和Rotation 重置为(0,0,0)。

### 模拟运行

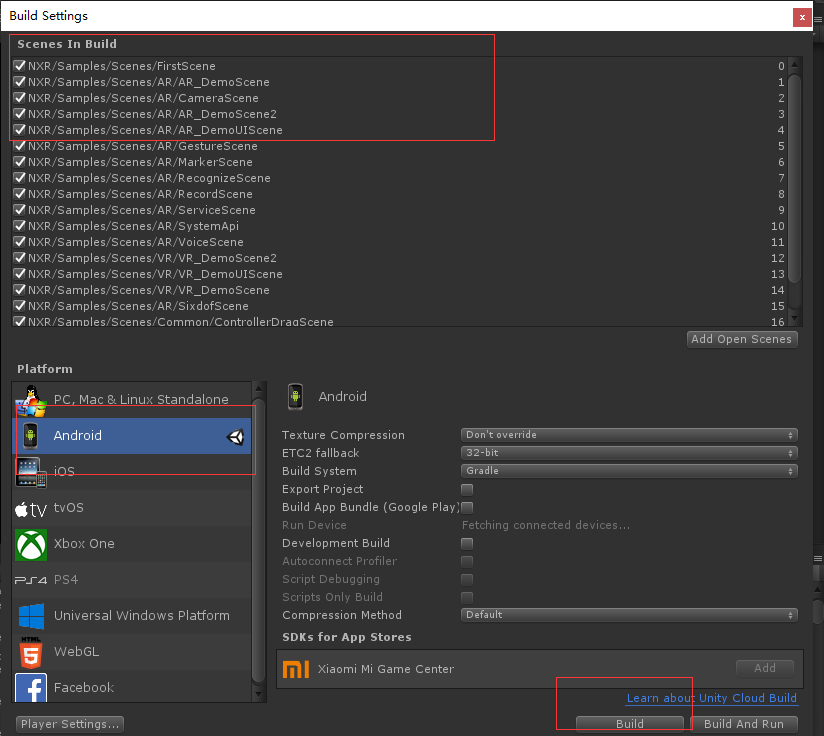
点击编辑器运行按钮，在Game 窗口可看到如下界面：



### 打包

打包前，检查当前选择的目标设备类型，需要选择的插件。

编辑器顶部工具栏找到HoloeverMR，进入XR Settings/Plugin Manager 后根据需要调整相关配置。最后进入File->Build Settings…，添加场景，平台切换到Android，点击Build 生成APK，安装到一体机。



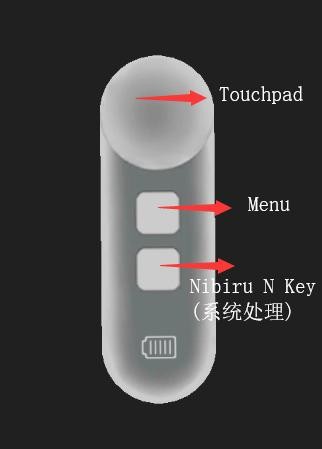
### 按键手柄交互

* 1. 一体机按键

按键键值对应表：

|  |  |  |
| --- | --- | --- |
| 键值描述 | Android 键值 | Unity 键值 |
| 确认 | 23/66 | 10 or  KeyCode.JoystickButton0 |
| 上 | 19 | KeyCode.UpArrow |
| 下 | 20 | KeyCode.DownArrow |
| 左 | 21 | KeyCode.LeftArrow |
| 右 | 22 | KeyCode.RightArrow |
| 返回 | 4 | KeyCode.Escape |
| 功能键F1 | 57 | FunctionKeyCode.NF1 |
| 功能键F2 | 130 | FunctionKeyCode.NF2 |

## 手柄按键



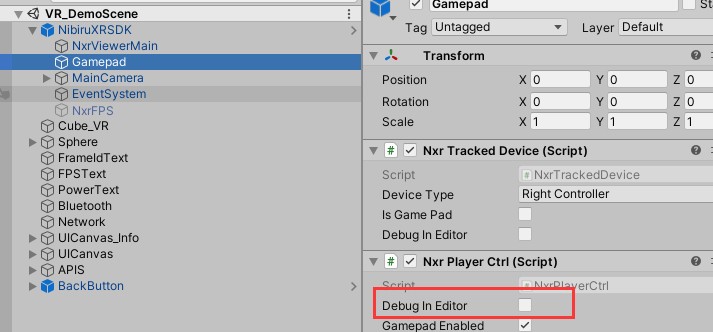
Touchpad 中Touch 坐标范围：

X 方向，左 -> 右（-1~1），Y 方向，上 –> 下（-1~1）

// 手柄触摸板，触摸坐标

Vector2 pos = InteractionManager.TouchPadPosition;

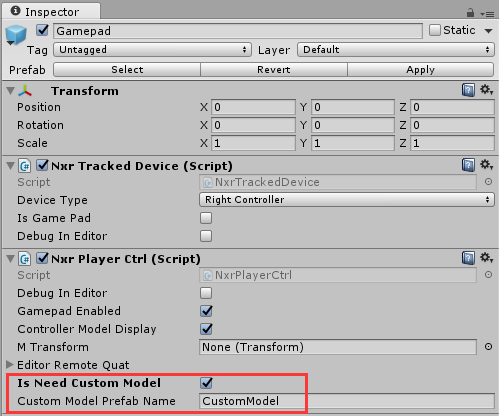
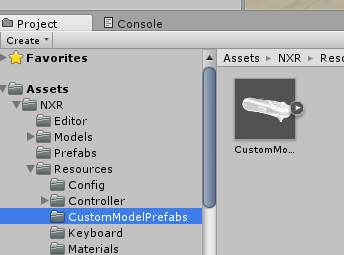
在编辑器下，可以在运行时，选中Gamepad，勾选NxrPlayerCtrl 里面的DebugInEditor，场景内会加载 3dof 模型。开发者可以进行简单模拟调试。



在新版本中，SDK 会根据系统配置自动加载匹配的手柄模型，如果开发者想自己修改手柄模型（3dof、6dof 均支持），需将自定义的手柄模型做成预制体并放在Assets/Resources/CustomModelPrefabs 目录下（预制体制作参考该目录下的CustomModel.prefab），选中Gamepad，勾选NxrPlayerCtrl 里面的IsNeesCustomModel 开启自定义手柄模型功能，设置CustomModelPrefabName（预制体名称）。

【示例场景：NXR/Samples/Common/CustomCtrlScene】

注：如果需要修改的场景过多，可直接在NxrPlayerCtrl.cs 脚本里修改IsNeesCustomMode 和CustomModelPrefabName 这两个参数。



开发者可通过下面的接口获取相应按键状态：

|  |  |
| --- | --- |
| 一体机按键按下 | NxrInput.GetKeyDown(int keycode) |
| 一体机按键长按 | NxrInput.GetKeyPressed(int keycode) |
| 一体机按键抬起 | NxrInput.GetKeyUp(int keycode) |
| 3DOF 手柄按键按下 | NxrInput.GetControllerKeyDown (int keycode) |
| 3DOF 手柄按键长按 | NxrInput.GetControllerKeyPress ed(int keycode) |
| 3DOF 手柄按键抬起 | NxrInput.GetControllerKeyUp(in t keycode) |
| Nolo 手柄按键按下 | NxrInput.GetControllerKeyDown (int keycode, int  NACTION\_HAND\_TYPE type) |
| Nolo 手柄按键长按 | NxrInput.GetControllerKeyPress ed(int keycode, int  NACTION\_HAND\_TYPE type) |
| Nolo 手柄按键抬起 | NxrInput.GetControllerKeyUp(in t keycode, int  NACTION\_HAND\_TYPE type) |

键值keycode：

|  |  |
| --- | --- |
| CKeyEvent.  KEYCODE\_CONTROLLER\_TOUCHPAD\_TOUCH | 手柄触摸板触摸  事件键 |
| CKeyEvent.KEYCODE\_CONTROLLER\_TRIGGER | 6DOF 手柄扳机键 |
| CKeyEvent.KEYCODE\_CONTROLLER\_MENU | 手柄菜单键 |
| CKeyEvent.KEYCODE\_CONTROLLER\_TOUCHPAD | 手柄触摸板按键 |
| CKeyEvent.KEYCODE\_CONTROLLER\_VOLUMN\_DOWN | 手柄音量减 |
| CKeyEvent.KEYCODE\_CONTROLLER\_VOLUMN\_UP | 手柄音量加 |
| CKeyEvent.KEYCODE\_3DOF\_CONTROLLER\_TRIGGER | 3DOF 手柄板机键 |

### API 接口函数一览

* 1. 相机 FOV 参数调节

|  |  |
| --- | --- |
| 功能 | 函数名及调用方式 |
| 更新左右眼相机FOV，范围  [5,80] | NxrViewer.Instance.UpdateCameraFov(4  0); |
| 恢复FOV | NxrViewer.Instance.ResetCameraFov(); |
| 更新相机远裁剪面距离（Start()  中调用） | NxrViewer.Instance.UpdateCameraFov(2  000f) |

* 1. 一体机性能等级参数

|  |  |
| --- | --- |
| 功能 | 函数名及调用方式 |
| 不同一体机性能等级（具体可通过NxrGlobal.PERFORMANCE 区分，LOW,NORMAL,HIGH 共  3 档） | NxrGlobal.platPerformanceLevel |

* 1. 启动画面配置参考

|  |  |
| --- | --- |
| 功能 | 函数名及调用方式 |
| 应用启动分屏画面参  考 | NXR/Resource/目录提供了参考启动图 |

* 1. 软件盘输入控件



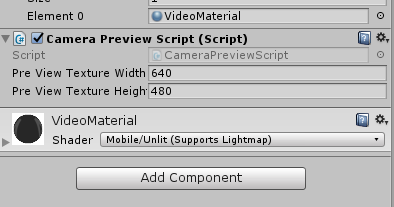
支持字母大小写，数字，标点符号。

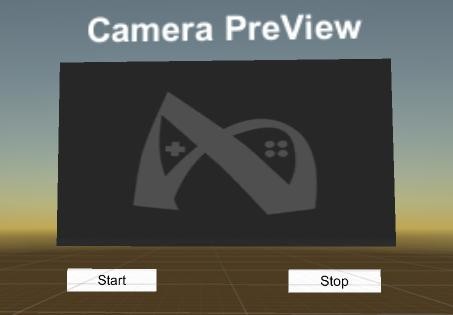
|  |  |
| --- | --- |
| 功能 | 函数名及调用方式 |
| 设置输入的字符串需要显  示在哪个Text 组件附近 | HoloeverKeyBoard.Instance.SetText(text); |
| 获取键盘的Transform | HoloeverKeyBoard.Instance.GetKeyBoardTransform(); |
| 获取键盘当前输入的字符  串 | HoloeverKeyBoard.Instance.GetKeyBoardString(); |
| 显示键盘，pageIndex=0  显示字母，1 显示数字 | HoloeverKeyBoard.Instance.Show(0, new Vector3(0, -  0.3f, 1), new Vector3(30, 0, 0)); |
| 显示键盘使用默认位置 | HoloeverKeyBoard.Instance.Show(); |

## 获取头部姿态数据

|  |  |
| --- | --- |
| 功能 | 函数名及调用方式 |
| 获取当前头部姿态数据 | NxrViewer.Instance.GetHead().transform |

* 1. 相机预览画面

【示例场景：NXR/Samples/Scenes/AR/CameraScene】



|  |  |
| --- | --- |
| 功能 | 函数名及调用方式 |
| 注册状态回调 | HoloeverService.OnCameraIdle += CameraIdle; HoloeverService.OnCameraBusy +=  CameraBusy; |
| 查询状态 | int cameraId = 1; （不同机器硬件摄像头可能不一样，0=一般是后置摄像头，  1= 一 般 是 前 置 摄 像 头 ） NxrViewer.Instance.GetHoloeverService().GetCa  meraStatus(cameraId); |
| 调用预览画面 | // 1=使用前置摄像头（头戴端），0=使用后置摄 像 头 （ 主 机 端 ） NxrViewer.Instance.GetHoloeverService().StartC  ameraPreView(cameraId); } |
| 暂停预览画面 | holoeverService.StopCamereaPreView(); |

在示例场景内，相机预览的画面显示在Panel 上面，且GameObject 下面需要挂载CameraPreviewScript.cs，可以修改预览纹理的宽高大小，尽量不要太 大，过大对性能有影响，宽高比例最好保持 16：9；

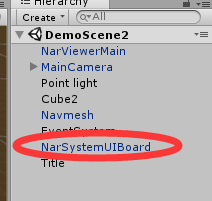
注：如果遇到画面为黑色，可以尝试修改cameraId。

使用此接口，确保硬件支持摄像头。如果使用过程中遇到问题时，可以查看运行的LOG，或者查看 Assets\Plugins\Android\ AndroidManifest.xml 下是否有camera 权限：

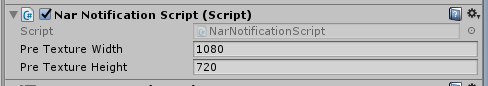
<uses-permission android:name="android.permission.CAMERA"/> 默认SDK 已经添加了此权限。

## 系统通知栏

【示例场景：Samples/Scenes/AR/AR\_DemoScene2】



将NXR/Prefabs/NxrSystemUIBoard 预制体拖到场景内，调节transform 到合适位置。



调节纹理宽高，注意width/height 比例最好和transform.scale 的 x/y 比例保持一致。否则可以出现拉伸或压扁情况。

注：此特性需要系统支持，如果无显示请尝试更新系统版本。系统版本号必须大于等于 1.12.XXX

## 瞄准点选中控制

【示例场景：NXR/Samples/Scenes/FirstScene】

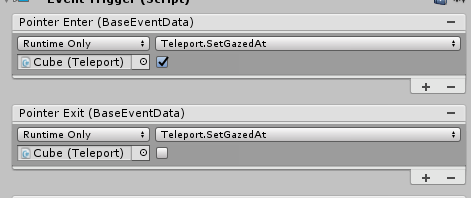
|  |  |
| --- | --- |
| 功能 | 函数名及调用方式 |
| 系统白点 | NxrViewer.Instance.HeadControl =  HeadControl.GazeSystem; |
| APP 白点 | NxrViewer.Instance.HeadControl =  HeadControl.GazeApplication; |
| APP 白点Hover | NxrViewer.Instance.HeadControl =  HeadControl.GazeHover; |

|  |  |
| --- | --- |
| 功能 | 函数名及调用方式 |
| 系统白点大小调节 | NxrViewer.Instance.GazeApi(GazeTag.Set\_Size,  ((int) GazeSize.Larger).ToString()); |
| 系统白点颜色调节 | NxrViewer.Instance.GazeApi(GazeTag.Set\_Color,  "0.043f\_0.435f\_0.043f"); RGB 范围（0~1） |
| 系统白点可见性调节 | NxrViewer.Instance.GazeApi(GazeTag.Hide); |

系统白点与APP 白点区别主要在于，APP 白点由于插帧的影响会出现抖动，而系统白点不存在此问题。

场景内的物体比如 cube 可以被瞄准到了，但是我们怎么去监听这个瞄准事件呢？在 SDK 内提供了INxrGazeResponder.cs 接口，通过实现这个接口你可以监听到瞄准进入、退出、确认等事件。

在Demo 下面，我们已经提供了一个Teleport.cs 示例，将其挂载到 cube 物体上面，同时为 cube 添加Event Trigger 组件，在Event Trigger 下面新增Pointer Enter,Pointer Exit,Pointer Click 事件类型。





如上面 cube 被选中时，就会更换其颜色。cube 选中后，按下OK 键会触发cube 随机更换位置。

针对Canvas 里面 Button 组件点击事件，可以直接在相对应的On Click 事件内添加处理逻辑即可。

## 按键功能接口

【示例场景：NXR/Samples/Scenes/Common/InputKeyScene】

|  |  |
| --- | --- |
| 功能 | 函数名及调用方式 |
| 一体机按键 | NxrInput.GetKeyDown(int keycode) NxrInput.GetKeyPressed(int keycode)  NxrInput.GetKeyUp(int keycode) |
| 3DOF 手柄按键 | NxrInput.GetControllerKeyDown(int keycode) NxrInput.GetControllerKeyPressed(int keycode)  NxrInput.GetControllerKeyUp(int keycode) |
| Nolo 手柄按键 | NxrInput.GetControllerKeyDown(int keycode, int NACTION\_HAND\_TYPE type) NxrInput.GetControllerKeyPressed(int keycode, int NACTION\_HAND\_TYPE type) NxrInput.GetControllerKeyUp(int keycode,  int NACTION\_HAND\_TYPE type) |

## 视角锁定和重置接口

|  |  |
| --- | --- |
| 功能 | 函数名及调用方式 |
| 视角锁定 | NxrViewer.Instance.LockHeadTracker=true; |
| 视角重置 | NxrViewer.Instance.Recenter(); |

* 1. 纹理质量

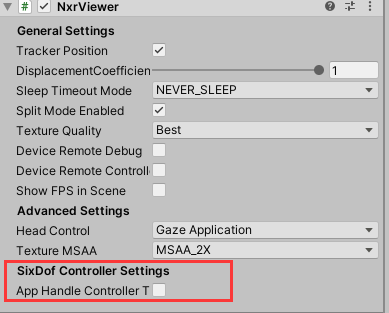
在NxrViewer 下面可以看到 Texture Quality 选项，默认Good（画面质量正常，无模糊现象），其他 2 个选项分别为 Simple（画面质量最低，会有模糊现象），Best（画面质量最高，最清晰）。

注：调整最好在NXR/Prefabs/NxrViewerMain 中修改。在单独场景修改适合于编辑器运行。

## 6DOF 手柄支持

如果连接了 6dof 双手柄，SDK 会自动加载模型显示，进行交互。

默认手柄Trigger 按键在 SDK 内会被当做确认键使用，如果当前应用需要使用Trigger 按键，可在NxrViewer 配置中，勾选 App Handle Controller Trigger Event 接收Trigger 键响应。



## 系统相关接口

|  |  |
| --- | --- |
| 功能 | 函数名及调用方式 |
| 系统分屏：适用于整个场景为2D UI,此时可以调用系统分屏，  而不使用我们的 SDK 分屏 | NxrViewer.Instance.SetSystemSplitMode(flag);  （flag=1 系统将当前场景划分为左右两屏，flag=0 应用分  屏即挂载我们的相机脚本产生的分屏效果） |
| 获取Android SD 卡路径返回路径，如：  /storage/emulated/0 | NxrViewer.Instance.GetStoragePath(); |
| 虚拟鼠标注册服务，仅适用于普  通分屏模式下的 2D 应用使用 | NxrViewer.Instance.GetHoloeverService().  RegisterVirtualMouseService |
| 虚拟鼠标注销服务 | NxrViewer.Instance.GetHoloeverService().  UnRegisterVirtualMouseService |
| 虚拟鼠标控制 | NxrViewer.Instance.GetHoloeverService().  SetEnableVirtualMouse(bool enabled) |
| 获取蓝牙状态，网络状态：0=  无网络，1=有网络 | HoloeverTaskApi.GetNetworkStatus() |
| 获取网络状态，蓝牙状态，0=  关闭，1=开启 | HoloeverTaskApi.GetBluetoothStatus() |
| 设置IPD (Start 中调用或按键触  发等) | NxrViewer.Instance.SetIpd(0.060f); |
| 获取IPD | NxrViewer.Instance.GetIpd(); |
| 获取MAC 地址 | HoloeverTaskApi.GetMacAddress(); |
| 获取设备唯一标识 | HoloeverTaskApi.GetDeviceId(); |
| 根据包名启动某个应用 | HoloeverTaskApi.LaunchAppByPkgName("com.nibiru.vr.l  ib2.test"); |

HoloeverService holoeverService = NxrViewer.Instance.GetHoloeverService();

|  |  |
| --- | --- |
| 功能 | 函数名及调用方式 |
| 获取驱动板软件版本号 | holoeverService .GetVendorSWVersion() |
| 获取光线传感器数值：（尽量不  要频繁调用） | holoeverService .GetLightValue() |
| 获取距离传感器数值：（尽量不  要频繁调用） | holoeverService .GetProximityValue() |
| 获取屏幕亮度值 | holoeverService .GetBrightnessValue() |
| 调节屏幕亮度 | holoeverService .SetBrightnessValue(holoeverService  e.GetBrightnessValue() - 1); |
| 2D/3D 显示模式 | holoeverService .GetDisplayMode() |
| 切换 2D/3D 显示模式 | holoeverService .SetDisplayMode(DISPLAY\_MOD  E.MODE\_3D); |
| 隐藏/显示触摸板光标 | HoloeverService .SetEnableTouchCursor(false); |
| 处理回调的 Sensor 数据 | HoloeverService .OnSensorDataChangedHandler  += onSensorDataChanged;  注册接受加速度，陀螺仪，地磁数据回调holoeverService .RegisterSensorListener(SENSOR  \_TYPE.ACCELEROMETER,SENSOR\_LOCATION. CONTROLLER); SENSOR\_LOCATION.CONTROLLER 为获取主  机端 Sensor 数据SENSOR\_LOCATION.HMD 为获取眼镜端  Sensor 数据取消停用接口  holoeverService .UnRegisterSensorListener(); |
| 获取设备唯一标识 | HoloeverTaskApi.GetDeviceId(); |
| 根据包名启动某个应用 | HoloeverTaskApi.LaunchAppByPkgName("com.  nibiru.vr.lib2.test"); |

## Pro 版本高级服务

注：此处所说插件跟unity插件没有任何相关性，本文档中所说的插件就代表6DOF位移等功能接口。Pro SDK为付费SDK，必须购买了相应的license才可使用相关功能。

【最低系统版本要求】

设置-系统-关于设备-版本类型 | 版本号

版本类型：MR0001 ，版本号必须大于等于Holoever 1.10.XXX

## 授权相关说明

联系中科永光客服人员,提交要生成的包名。

步骤1: 将要申请使用的包名提交中科永光科技有限公司.

步骤2: 将获取后的 NibiruSDKKey.bin 复制覆盖在此路径下 Assets\\Plugins\\Android\\assets.\n"

步骤3: 将获取到的KEY的值,输入并提交.

同时将对应路径下的NibiruSDKKey.bin 拷贝到Assets/Plugins/Android/assets 目录下。

Step4：点击 unity 编辑器顶部HoloeverXR 标签，选择 SDK Verify 进入密钥配置界面，按照对应步骤，放置NibiruSDKKey.bin 文件，粘贴密钥字符串，最后 点 击 确 认 就 可 以 配 置 完 成 。 Step5：密钥配置完成后就可以进行打包操作。

## 插件验证结果

通过NxrGlobal.verifyStatus 可获取到当前的验证状态，如果失败，当前画面会冻住无法转动，且会一直输出" Verify

Failed "LOG。如果验证成功，就可以正常使用各个插件功能了。

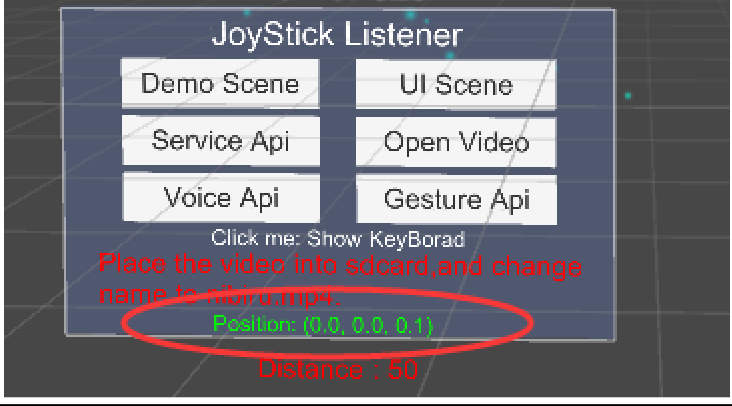
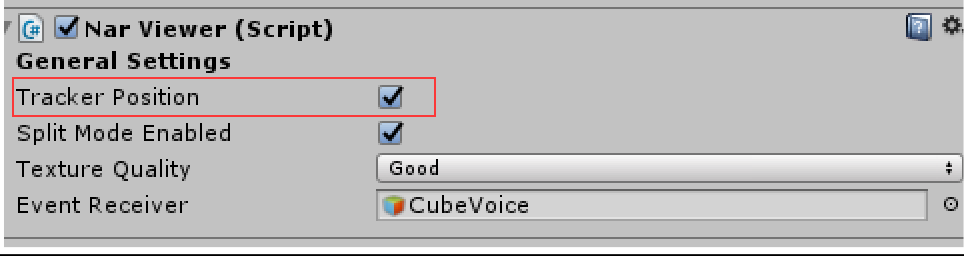
# 插件支持状态

因为硬件的差异性，插件在某些机器可能无法被支持，比如设备没有 Camera 或麦克风。此时需要通过获取插件的支持状态来判断当前硬件是否支持。

|  |  |
| --- | --- |
| 功能 | 函数名及调用方式 |
| 获取服务连接状态（服务连接  后，就可以查看插件支持状态 ） | NxrViewer.Instance.serviceReadyUpdatedDel egate += new NxrViewer.ServiceReadyUpdatedDelegate(O  nServiceReady); |
| 判断插件是否声明 | HoloeverTaskApi.IsPluginDeclared(Nxr.Internal.  PLUGIN\_ID.SIX\_DOF) |
| 判断插件是否被支持 | HoloeverTaskApi.IsPluginSupported(Nxr.Interna  l.PLUGIN\_ID.SIX\_DOF) |

注：在实际开发中建议开发者首先验证下插件支持状态，确保机器硬件支持。

# 6DOF 位移

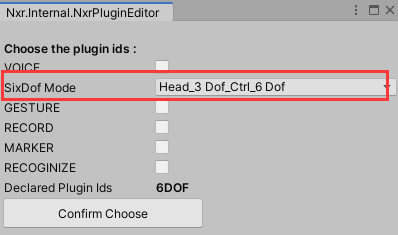
【示例场景：NXR/Samples/AR/AR\_DemoScene】NXR/Samples/Scripts/Pro/Holoever6DofTest.cs

UI 面板界面下方绿色文本Position:0,0,0 代表当前位移信息(x,y,z)

在NxrViewerMain 脚本上General Settings 下面Tracker Position 选项代表是否启用位移功能。

如果需要关闭，在脚本中调用NxrViewer.Instance.TrackerPosition=false 或者修改NXR/Prefabs/NxrViewerMain 预制体将选项关闭。

|  |  |
| --- | --- |
| 功能 | 函数名及调用方式 |
| 监听位移信息的变化 | NxrViewer.onSixDofPosition += OnHeadPosition;  void OnHeadPosition(float x, float y, float z) NxrViewer.onSixDofPosition -=  OnHeadPosition; |



针对 6DOF，在插件选择界面提供了 3 种模式，

头部 3dof 手柄 3dof，头部 3dof 手柄 6dof（默认选择），头部 6dof 手柄6dof。

注：6dof 的位移全部为绝对位移。

# 视频录制

【示例场景：NXR/Samples/AR/RecordScene】NXR/Sample/Scripts/Pro/RecordTestScript.cs

视频录制会将当前场景画面与相机画面进行叠加，输出到 mp4 文件上。

.支持 480p,720p,1080p 3 种分辨率大小

.支持设置前置或后置 camera

.支持设置视频码率大小

|  |  |
| --- | --- |
| 功能 | 函数名及调用方式 |
| 设置相机ID | HoloeverService.SetCaptureCamereaId(CAMERA  \_ID.FRONT); |
| 设置分辨率 | HoloeverService.SetCaptureVideoSize(VIDEO\_SI  ZE.V720P); |
| 设置mp4 文件输出路径 | string filePath =  NxrViewer.Instance.GetStoragePath() + "/unityrecord.mp4"; |
| 开始录制 | NxrViewer.Instance.GetHoloeverService().StartC  apture(filePath); |
| 暂停录制 | NxrViewer.Instance.GetHoloeverService().StopC  apture(); |

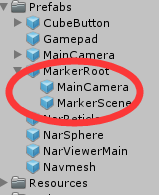
（可参考NXR/Sample/Scripts/Pro/RecordTestScript.cs）

注：

1. 录制的视频内容为场景内渲染的画面与相机预览画面叠加后的效果。
2. 录屏功能因为需要使用camera 相机，因此无法与手势识别和marker 识别等同样使用camera 相机的功能进行混合使用。

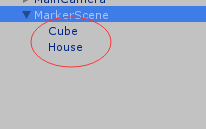
# Marker 识别

【示例场景：NXR/Samples/MarkerScene】



|  |  |
| --- | --- |
| 功能 | 函数名及调用方式 |
| 开始识别 | NxrViewer.Instance.GetHoloeverService().StartM  akerRecognize(); |
| 停止识别 | NxrViewer.Instance.GetHoloeverService().StopM  akerRecognize(); |
| 监听Maker 状态 | HoloeverMarker.OnMarkerFoundHandler = MarkerFound; HoloeverMarker.OnMarkerLostHandler =  MarkerLost; |

将Prefabs 目录下的MarkerRoot 预制体拖放到场景内。在MarkerScene 目录下，可以看到一个Cube 和一个房子。默认不显示，待识别marker 成功后，会自动触发显示。

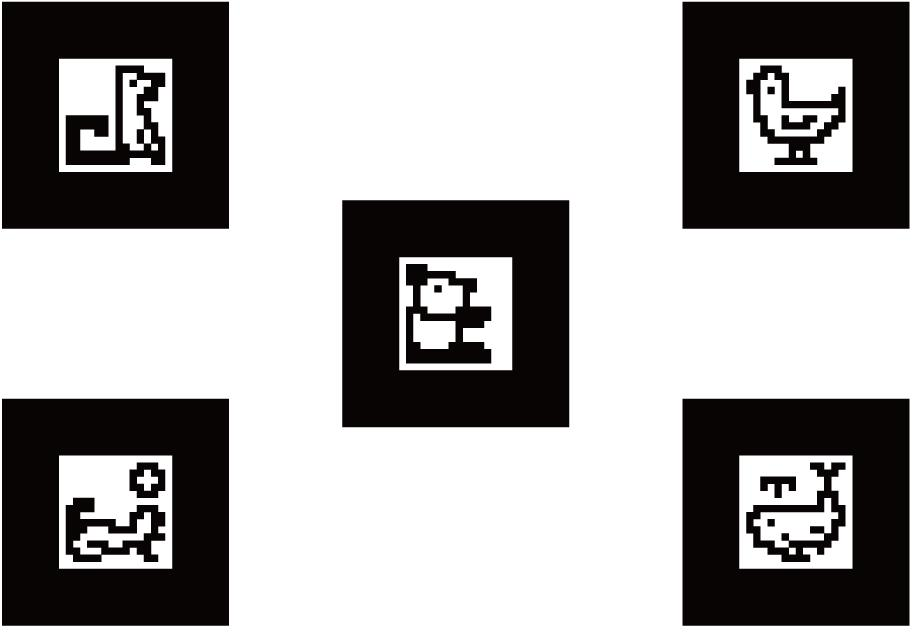


Marker 相关说明：

在Assets\Plugins\Android\assets\MarkerData 目录存放了marker 数据， 我们提供了一套默认的marker 配置文件（尽量不要随意修改），对应的主marker 图片在Assets\NXR\Resources\ 。

将Marker Guide/Marker 示例目录下面的图片打印到纸上。然后按照如下图所示进行排列组合（打印的尺寸和摆放的距离严格按照图示进行）。





注：

1. 开发者自己新增的marker 配置文件都需要放置在Assets\Plugins\Android\assets\MarkerData 目录下面。
2. marker 识别功能因为需要使用camera 相机，因此无法与手势识别和录屏等同样使用camera 相机的功能进行混合使用。

# 物体拖拽

【示例场景：NXR/Samples/Common/ControllerDragScene】

在需要拖拽的物体上添加NxrDragableItem.cs，新建脚本DragTest.cs 处理选中事件。

|  |  |
| --- | --- |
| 功能 | 函数名及调用方式 |
| 是否处于拖拽状态 | NxrDragableItem.IsDraging |
| 进入拖拽状态，物体跟随 | NxrDragableItem.OnBeginDrag(Transform  parent); |
| 离开拖拽状态，物体更新 | NxrDragableItem.OnEndDrag(Transform  parent); |

在DragTest.cs 里面，我们提供选中物体，按下确认键时，触发拖拽效果，然后就可以转动头部或手柄，将物体拖拽到指定位置，再次按下确认键将结束拖拽。

# 权限申请

|  |  |
| --- | --- |
| 功能 | 函数名及调用方式 |
| 申请权限 | NxrViewer.Instance.GetHoloeverService().Requs  etPermission(string[]) |

权限名称列表：

相 机 "android.permission.CAMERA";

外 部 存 储 写 "android.permission.WRITE\_EXTERNAL\_STORAGE"; 外部存储读 "android.permission.READ\_EXTERNAL\_STORAGE"; 位置 "android.permission.ACCESS\_COARSE\_LOCATION";

网络状态 "android.permission.ACCESS\_NETWORK\_STATE"; 设置 "android.permission.WRITE\_SETTINGS";

蓝牙 "android.permission.BLUETOOTH";

蓝牙管理 "android.permission.BLUETOOTH\_ADMIN"; 网络 "android.permission.INTERNET";

任务 "android.permission.GET\_TASKS";

录音 "android.permission.RECORD\_AUDIO";

电话状态 "android.permission.READ\_PHONE\_STATE";

如果遇到权限无法申请，或申请失败的情况。可以修改 Player Settings/Other Settings/Target API Level，选择API level 22 绕过权限问题。

如果当前系统不支持，SDK 会输出提示LOG：

\*\*\*\*\*\*\*[RequestPermission Warnning]\*\*\*\*\*\*\*

\*\*\*\*\*\*\*System Currently Not Support, Please Take Care !!!\*\*\*\*\*\*\* ……

# APK 加密功能

编辑器顶部工具栏找到HoloeverXR，进入APK Encryption 勾选 Enable APK Encryption，点击 Confirm 确认。

此加密功能需要配合加密工具使用：

如需加密,请联系中科永光客户服务

加密后的APK 只能在指定的一体机运行使用。

1. 通过加密工具生成加密文件，然后将文件放置到工程目录Assets/Plugins/Android/assets
2. 然后正常打包APK。

# 多线程渲染

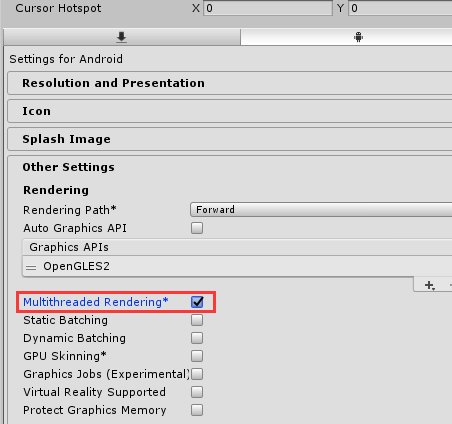
新版系统开始支持多线程渲染，在PlayerSettings-Other Settings 里勾选Multithreaded Rendering，然后打包运行。

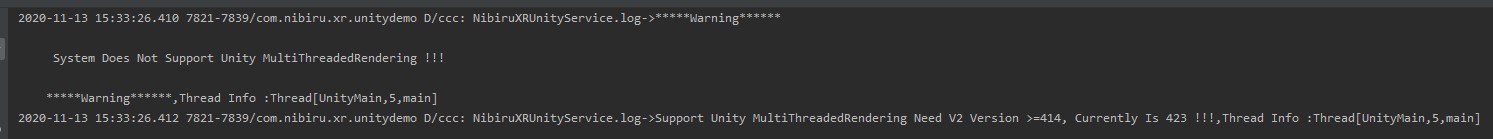
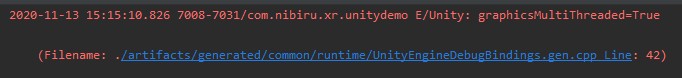
查看log，确认多线程渲染是否开启成功：graphicsMultiThreaded=（true： 多线程渲染已开启 false：未开启），

如果出现以下两条log：

1. \*\*\*\*\*Warning\*\*\*\*\*\*\n\n System Does Not Support Unity MultiThreadedRendering !!! \n\n\*\*\*\*\*Warning\*\*\*\*\*\*
2. Support Unity MultiThreadedRendering Need V2 Version >=414, Currently Is " + NxrGlobal.soVersion（当前 so 库版本） + " !!!

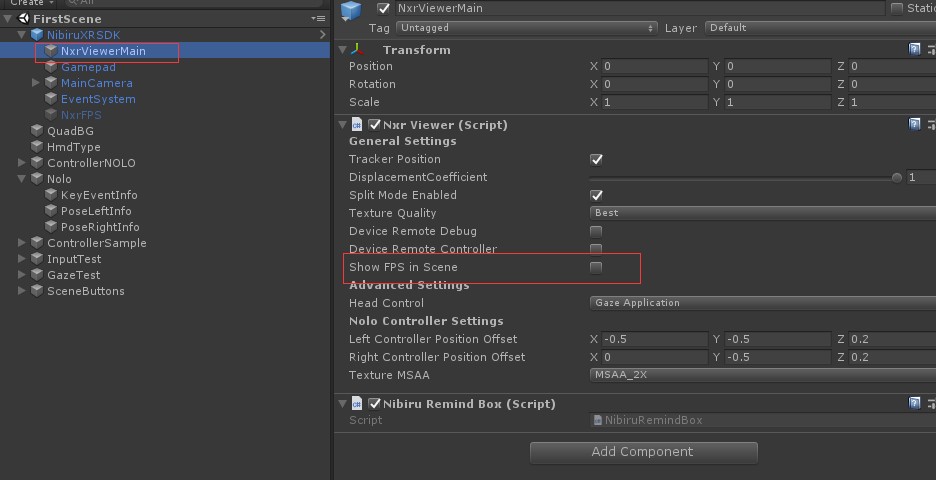
说明系统不支持多线程渲染,可能出现黑屏等现象。





### 辅助工具

显示帧率，HoloeverXRSDK/NxrViewerMain，勾选 Show FPS in Scene。



### 常见问题解答

* 1. 如果当前场景内包含 2D UI 且里面包含有正交相机，怎么进行分屏？ 我们建议将 2D UI 也放置到 3D 场景，可以参考上面（5）条目。其次整个场景最好只保留 1 个主相机。
  2. 为什么在编辑器显示分屏，可是在一体机运行没有分屏？

1. 检查AndroidManifest 是否加上了NVR 标签。
2. 检查脚本挂载的相机是不是MainCamera。
   1. 为什么在打包APK 时失败？
3. 检查Android 引用的jar 包是否有冲突。
4. 在编辑器查看Console 错误日志，具体分析原因。
   1. 为什么打包成功，但是运行崩溃或黑屏？
5. 打开Eclipse 的LogCat，通过Tag=Unity 过滤出Unity 相关Log，查看是否出现异常错误。
6. : I/Unity(4350): NullReferenceException

at UnityEngine.Material..ctor (UnityEngine.Shader shader) [0x00000]

遇到这个错误，打开Edit->Project Settings->Graphics， 将NAR/Resources 目录下的UnlitTexture.shader 和 SolidColor.shader 拖入右侧 Always Included Shaders 列表。

1. 检查Plugins/Android 目录下的jar 是否有重复。
2. 检查PlayerSetting 是否选择了Multithreaded Rendering 开关。如果开启了，请关闭后打包再运行。
   1. 帧率不稳定或者间隔一段时间掉帧，下面给出一些优化建议：
3. UGUI 的Image 组件的RaycastTarget 勾选以后会消耗一些性能， 为了节省性能就不要勾选它了，不仅Image 组件Text 组件也有这样的问题。 一般 UI 里也就是按钮才需要接收响应事件，那么大部分image 和text 是不需要开RaycastTarget 的。
4. 尽量少使用Update LateUpdate FixedUpdate，这样也可以提升性能和节省电量。多使用事件（不是 SendMessage，使用自己写的，或者 C#中的事件委托）。
5. 如果非必要情况下，尽量避免使用实时阴影，全局光照，以此提高运行效率。
   1. 要达到比较流畅的效果，对帧率及场景有什么要求？ 场景帧率不得低于 30 帧,建议 45 帧以上。
   2. 场景优化建议：
      1. 模型三角面片控制在单目 5 万以内 & 顶点数控制在 5 万以内
      2. 尽量减少或不使用Unity 灯光，采用Lightingmap 烘培光影信息。
      3. 尽量减少或不使用粒子系统。
   3. 帧率显示正常，画面卡顿

查看Unity Player Settings 相关配置：Blit Type 选择Never, Multithreaded Rendering 不要勾选