SeerLens™ One AR 眼镜

应用开发手册

上海诠视传感技术有限公司

Xvisio Technology (Shanghai) Co., Ltd.

Innovating machine perception capability beyond human capacity

www.xvisiotech.com | +86 21 5290 0903 | contact@xvisiotech.com

目录

一、开发环境搭建介绍

Unity 推荐使用 2020.3.14f1c1，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。

1.1Android 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

1.2URP 开发环境配置

软件名称	**软件版本**
Unity	2020.3.14
URP	7.1.8

1.3 开发注意事项

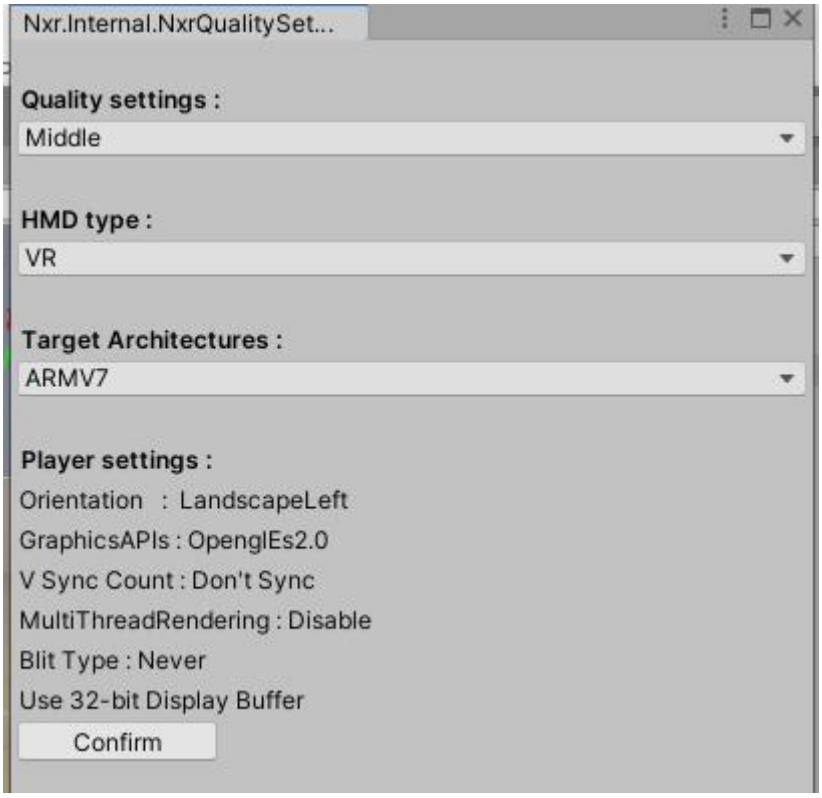
1.3.1Unity 注意事项

使用前请注意以下事项：

Unity 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 Nibiru XR - 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。

1.3.2Android 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"/>
```

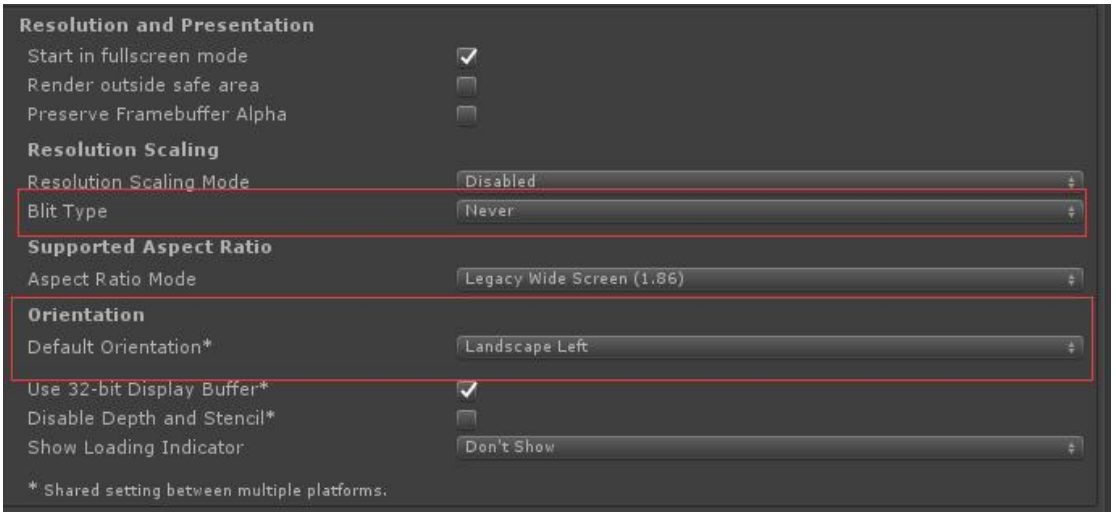
```
  <uses-permission
android:name="android.permission.ACCESS_COARSE_LOCATION" />
  <uses-permission
android:name="android.permission.WRITE_EXTERNAL_STORAGE"/>
  <uses-permission
android:name="android.permission.READ_EXTERNAL_STORAGE" />
  <uses-permission android:name="android.permission.INTERNET" />
  <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"/>
```

1.3.3SDK 推荐配置

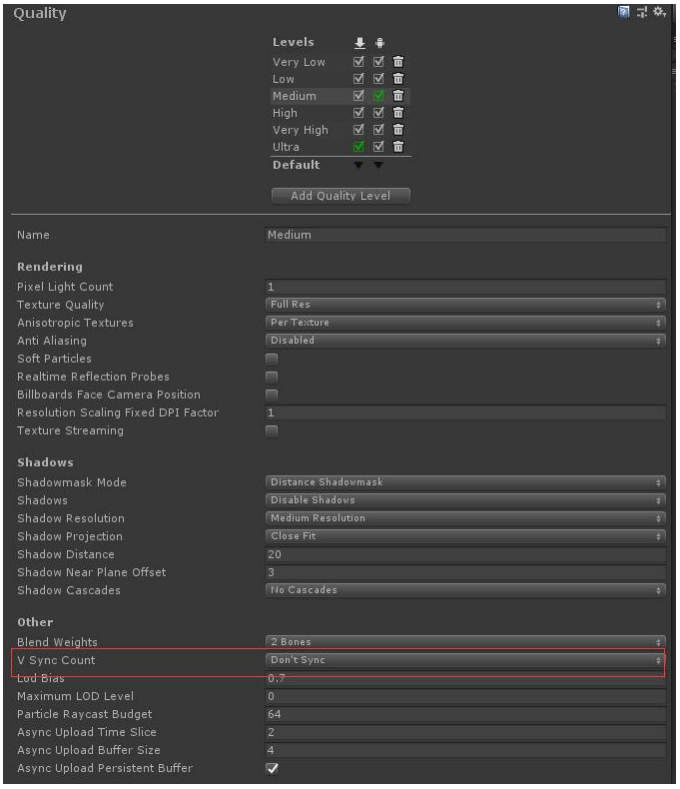
- Graphics APIS 不支持 Vulkan，对于 OpenGLES2,OpenGLES3，开发者根据自身需求自行选择。
- MultiThreaded Rendering 暂不支持，不需要勾选。
- Unity Player Setting 中 Default Orientation 需要选择 Landscape Left

  Resolution Scaling 下面的 Blit Type 需要选择为

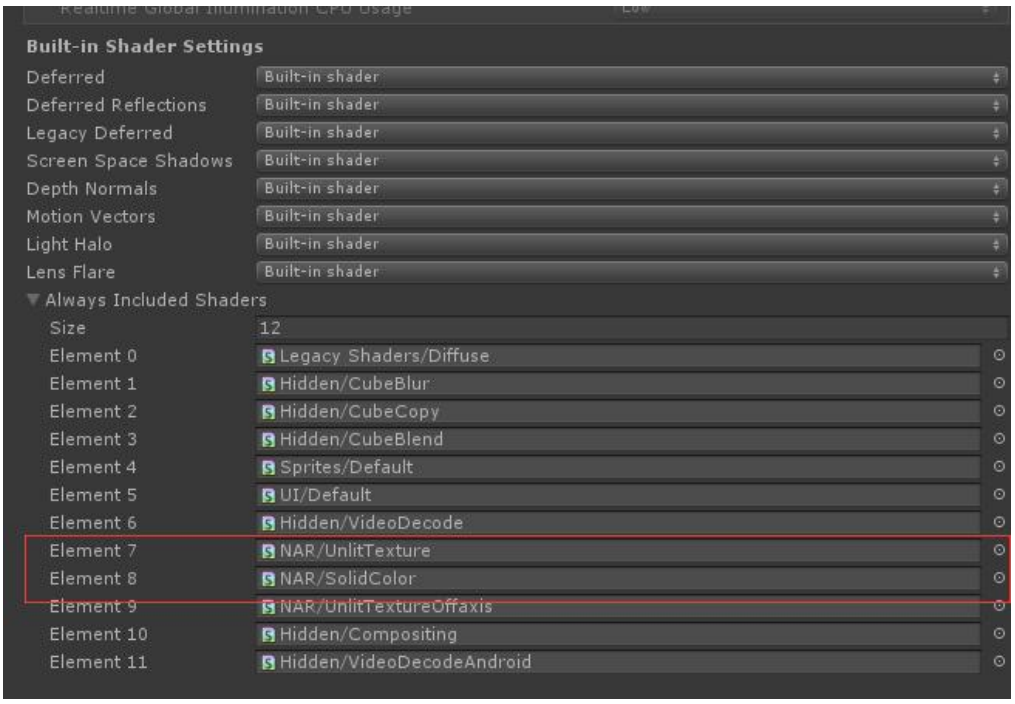
Never



- Quality 中的 V Sync Count 需要选择为 Don’ t Vsync



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



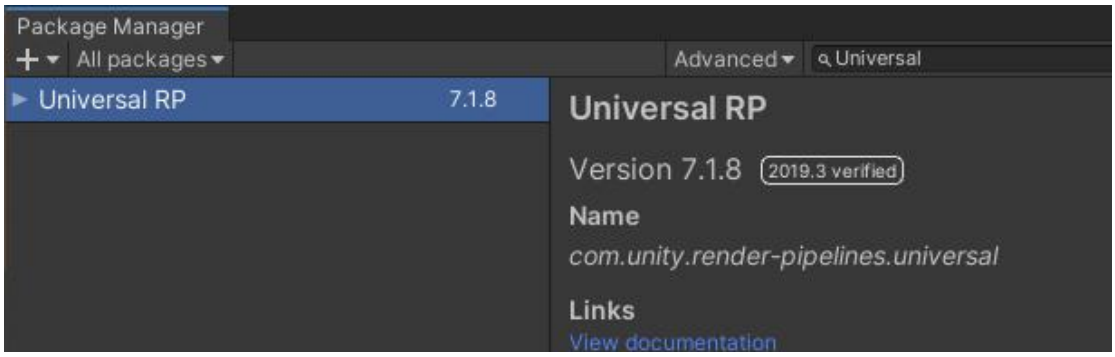
1.3.4Universal RP(通用渲染管线)配置说明

Unity 版本：**Unity2020.3.14**

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。

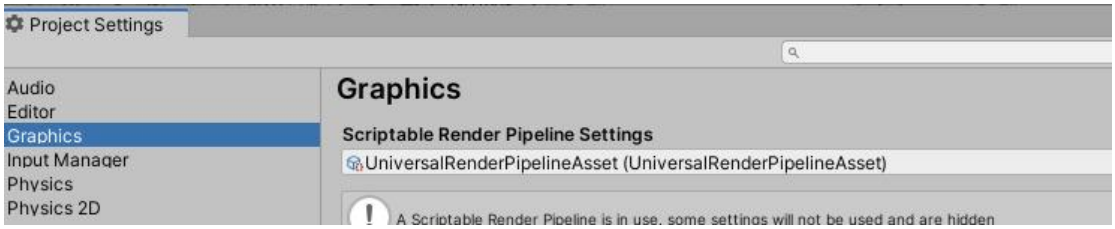
- 打开 Window/Package Manager，搜索 Universal RP，点击 Install。



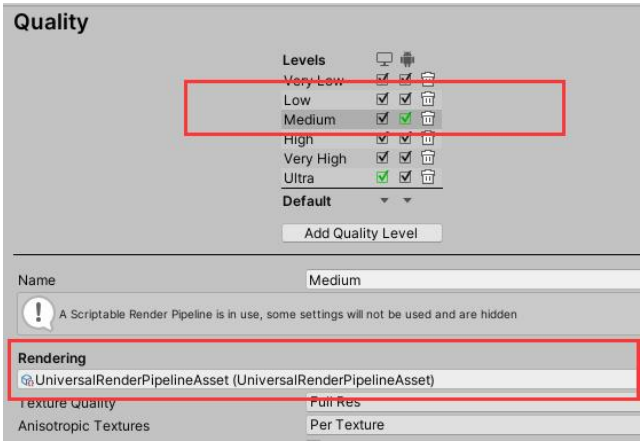
- 安装完成，创建 UniversalRenderPipelineAsset，

  Assets/Create/Remdering/Universal Render Pipeline/Pipeline Asset

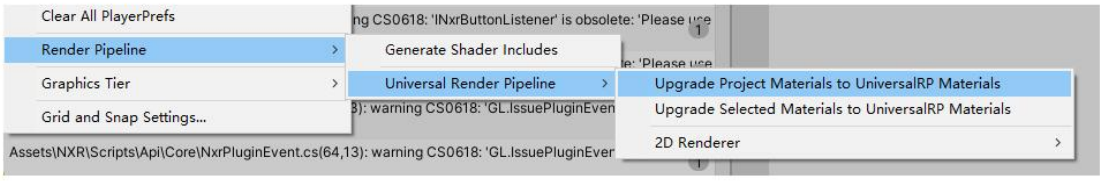
- 打开 Edit/Project Settings/Graphics/，在 Scriptable Render Pipeline Settings 中选择已创建的 PipelineAsset。



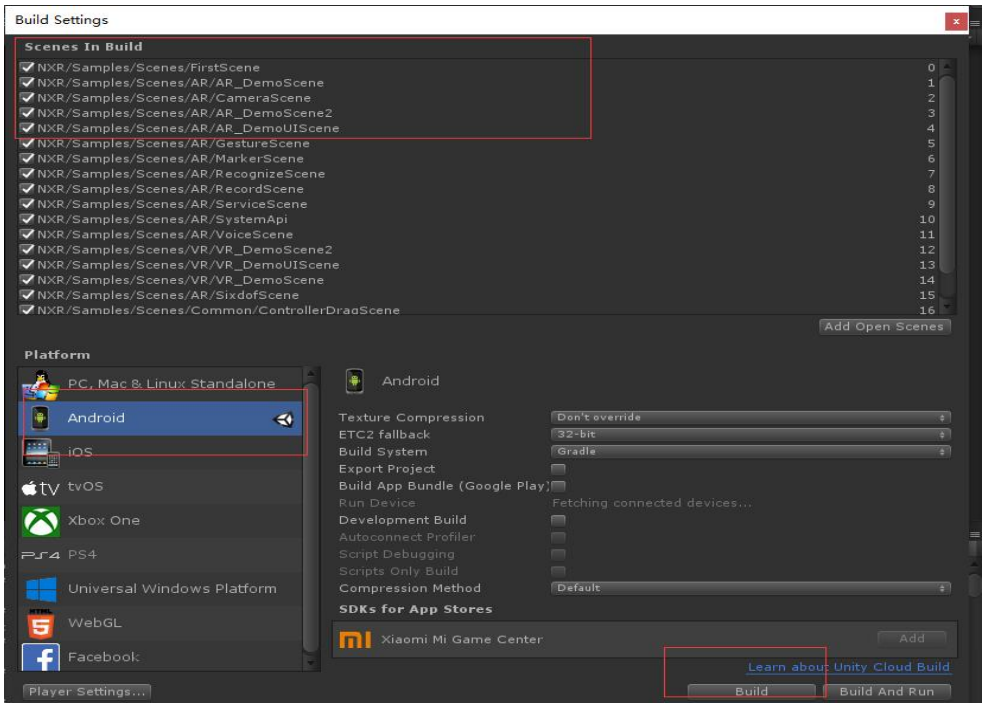
- 打开 Edit/Project Settings/Quality 界面，选择 Android 类别，在 Rendering 中选择已创建的 PipelineAsset。



- 升级工程材质



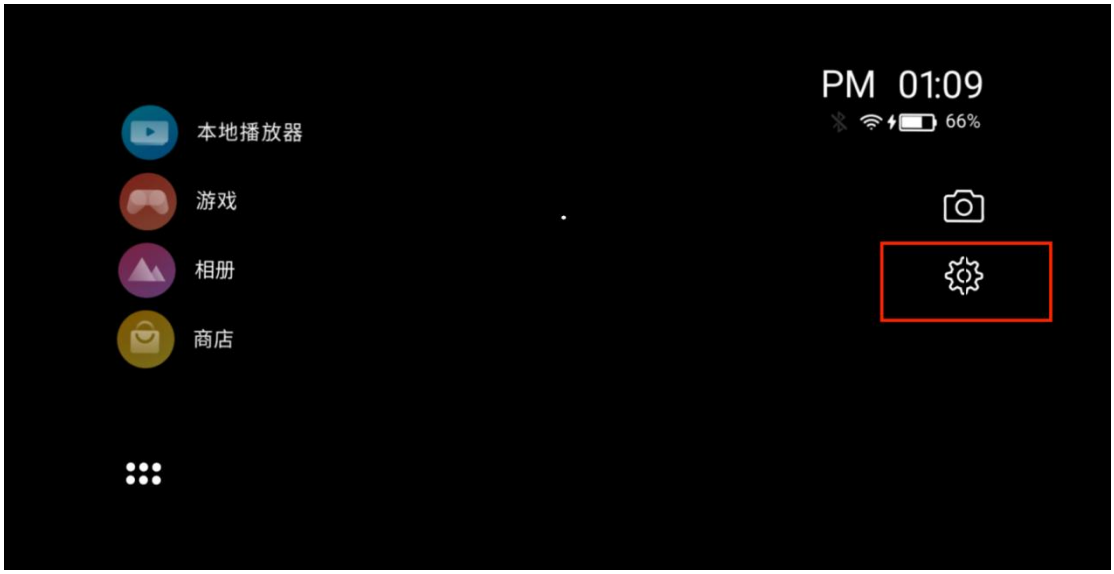
注：取消 UniversalRenderPipelineAsset 中的 SRP Batcher/Dynamic Batching 否则会出现内存泄露问题。



二、固件资料（Box）

2.1 主机更新

①打开设置

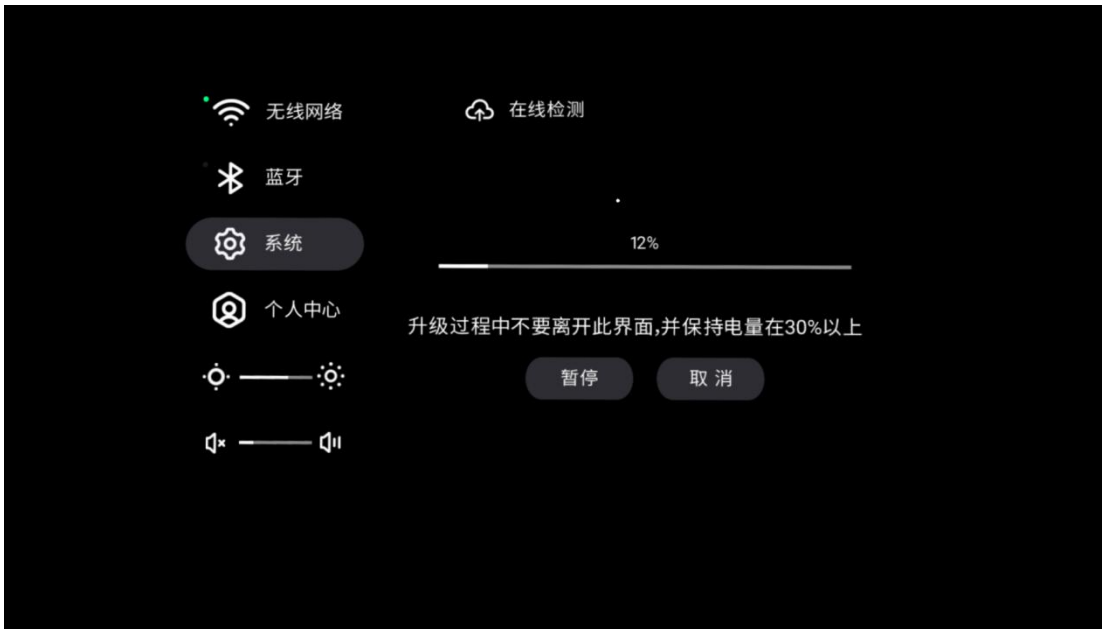


②左侧菜单点击系统，右侧选择到系统更新

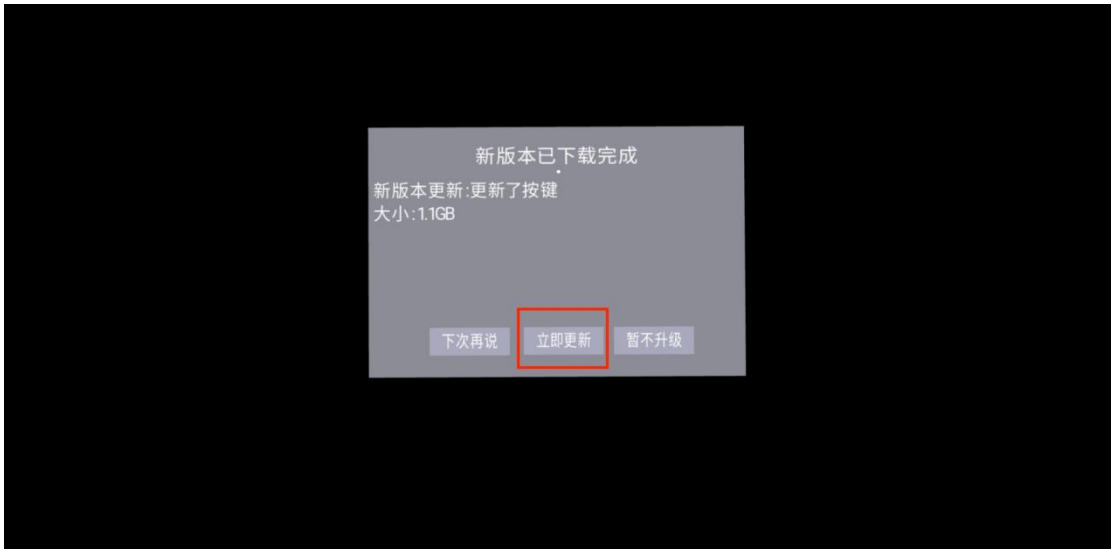
③选择在线升级

④检测到有升级包后点击确认

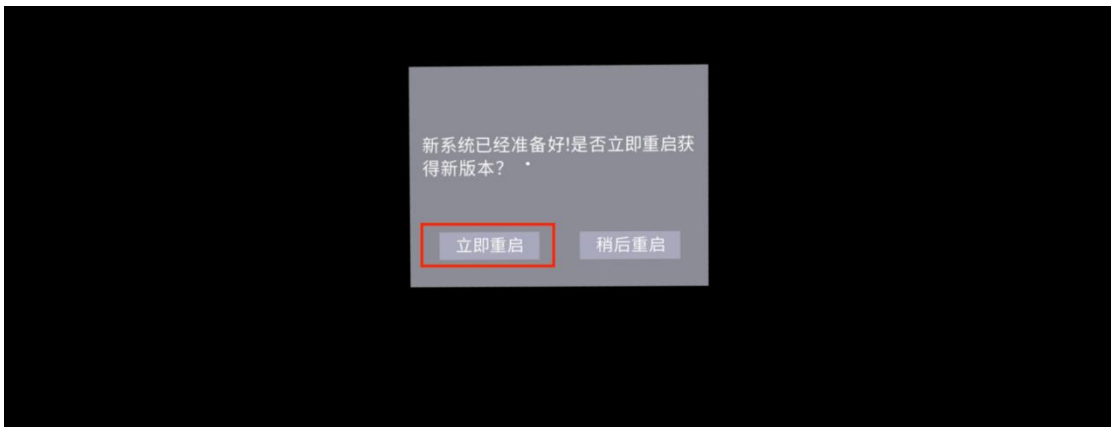
⑤确认后，系统自动下载升级包



⑥下载完成后点击立即更新



⑦等待升级完成，更新完成后会自动重启，重启后完成系统升级。



三、应用 sdk

3.1 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;

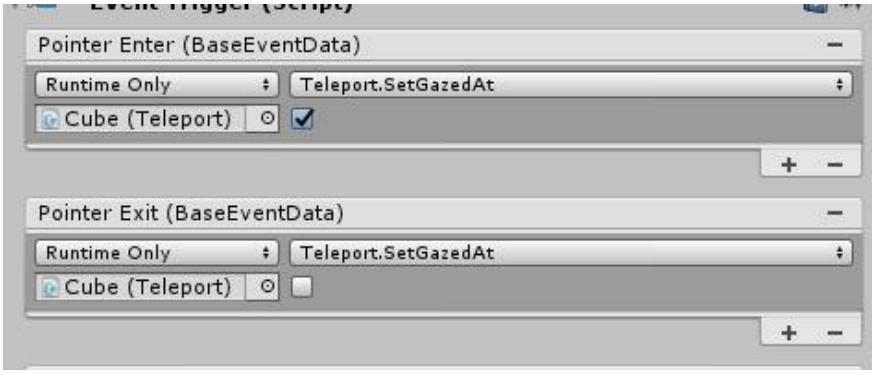
APP 白点 Hover	NxrViewer.Instance.HeadControl = HeadControl.GazeHover;

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。

3.2 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();

3.3 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。

3.4 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.GetNibiruService(). RegisterVirtualMouseService();
虚拟鼠标注销服务	NxrViewer.Instance.GetNibiruService(). UnRegisterVirtualMouseService();
虚拟鼠标控制	NxrViewer.Instance.GetNibiruService(). SetEnableVirtualMouse(bool enabled);
获取网络状态，网络状态：0=无网络，1=有网络	NibiruTaskApi.GetNetworkStatus(); NibiruTaskApi.IsWifiEnabled();

获取蓝牙状态，蓝牙状态，0=关闭，1=开启	NibiruTaskApi.GetBluetoothStatus(); NibiruTaskApi.IsBluetoothEnabled();
获取 MAC 地址	NibiruTaskApi.GetMacAddress();
获取设备唯一标识	NibiruTaskApi.GetDeviceId();
根据包名启动某个应用	NibiruTaskApi.LaunchAppByPkgName("com.nibiru.vr.lib2.test");
打开/关闭 WiFi	NibiruTaskApi.SetWifiEnable(bool enable);
打开/关闭蓝牙	NibiruTaskApi.SetBluetoothEnable(bool enable);
安装 apk	NibiruTaskApi.InstallApk(string apkPath);
安装 apk 成功/失败回调	NibiruTaskApi.SetInstallSuccessCallback(InstallSuccessCallback callback); NibiruTaskApi.SetInstallFailedCallback(InstallFailedCallback callback);
安装 apk 成功自动启动	NibiruTaskApi.InstallAndStartApk(string apkPath);
卸载 apk	NibiruTaskApi.UninstallApk(string packageName);
卸载 apk 成功/失败回调	NibiruTaskApi.SetUninstallSuccessCallback(UninstallSuccessCallback callback); NibiruTaskApi.SetUninstallFailedCallback(UninstallFailedCallback callback);

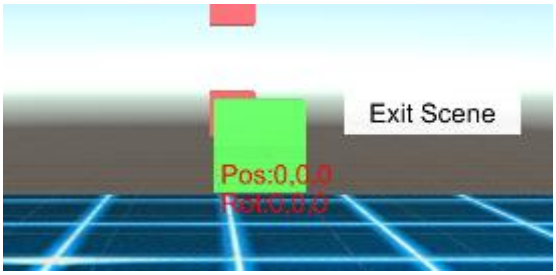
NibiruService nibiruService = NxrViewer.Instance.GetNibiruService();

功能	函数名及调用方式
获取驱动板软件版本号	nibiruService.GetVendorSWVersion();
获取光线传感器数值：（尽量不要频繁调用）	nibiruService.GetLightValue();
获取距离传感器数值：（尽量不要频繁调用）	nibiruService.GetProximityValue();
获取屏幕亮度值	nibiruService.GetBrightnessValue();
调节屏幕亮度	nibiruService.SetBrightnessValue(nibiruService.GetBrightnessValue() - 1);
隐藏/显示触摸板光标	nibiruService.SetEnableTouchCursor(false);
获取设备唯一标识	NibiruTaskApi.GetDeviceId();
根据包名启动某个应用	NibiruTaskApi.LaunchAppByPkgName("com.nibiru.vr.lib2.test");

3.5 6DOF 位移

【示例场景：NXR/Samples/Scenes/MainScenes/SixdofScene】

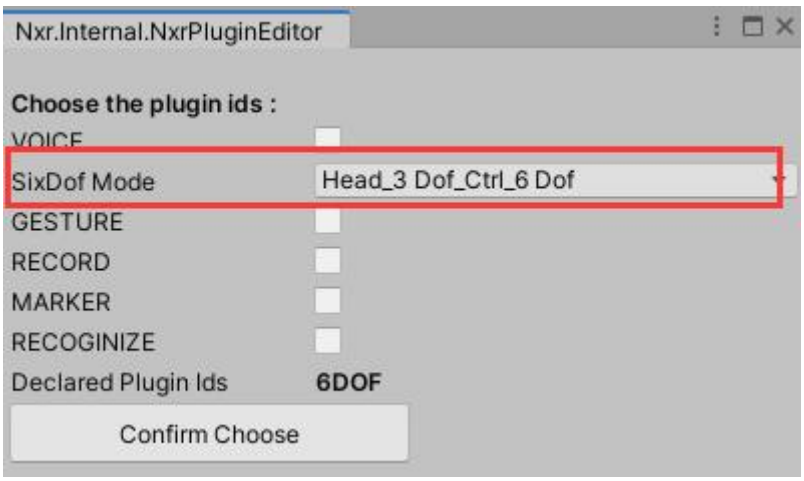
NXR/Samples/Scripts/MainScenes/SixdofTest



Pos: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。

3.6 权限申请

功能	函数名及调用方式
申请权限	NxrViewer.Instance.GetNibiruService().RequsetPermission(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]*******

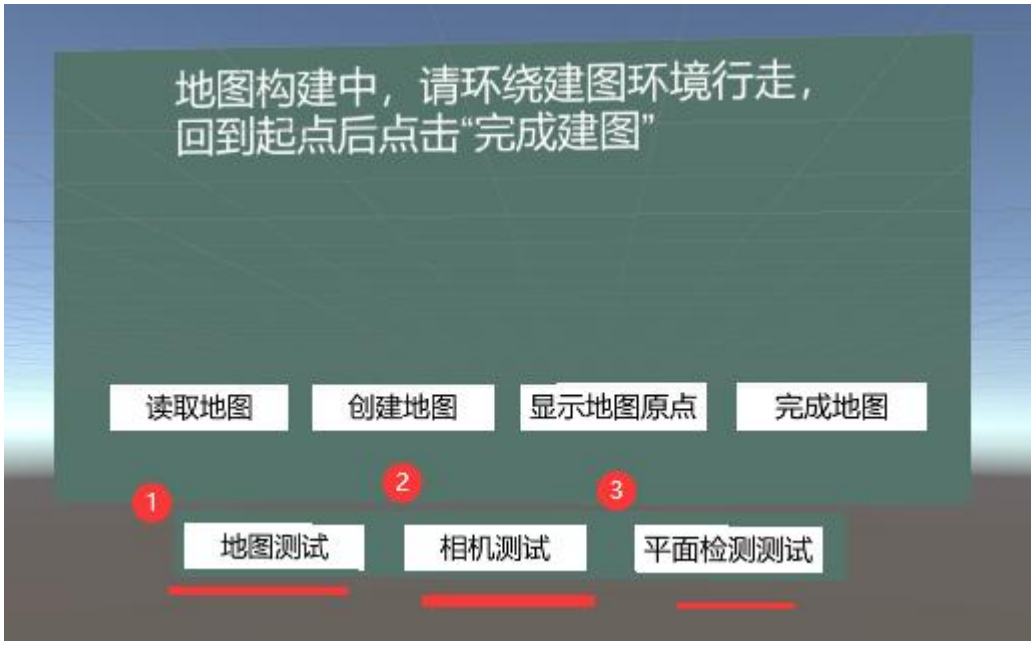
*******System Currently Not Support, Please Take Care !!!******* ……

3.7 SLAM 使用说明

3.7.1 SLAM 示例场景介绍

【示例场景：NXRSlam/Demo/SlamScene】

测试场景包含地图，相机，平面检测相关功能演示，通过下面按钮切换测试页面。



3.7.2 SLAM 接口函数一览

NxrSlam.Instance **函数名称**	**参数**	**返回值**	**描述**
Init	无	bool true 初始化成功	初始化

		false 初始化失败	
EnableNativeLog	bool enable		是否开启底层 log 打印
MapStatusAndQualityEvent	int quality 地图质量		监听地图状态变化和质量信息
MapPercentEvent	float percent 地图重合百分比		监听地图重合度
StreamDataEvent	支持 RGB，TOF，FISHEYE 相机数据回调。		监听相机数据回调
PlaneDetectionEvent	支持平面点和法向量的数据回调。		监听平面检测数据回调
UnInit	无		销毁
StartPlaneDetection	无	bool true 成功 false 失败	开始平面检测
StopPlaneDetection	无	bool true 成功 false 失败	暂停平面检测
StartBuildMap	无	bool	开始建图

		true 成功 false 失败	
StopBuildMap	string path 地图生成路径	bool true 成功 false 失败	暂停建图
StopSlam	无	无	暂停 Slam
StartSteaming	NxrSlam.RawDataType type 相机类型	无	开启相机流
StopStreaming	NxrSlam.RawDataType type 相机类型	无	关闭相机流
StartSlamWithMap	string path 地图生成路径	bool true 成功 false 失败	加载 path 路径对应的地图

Demo 示例脚本：

NXRSlam/Demo/SlamApi.cs

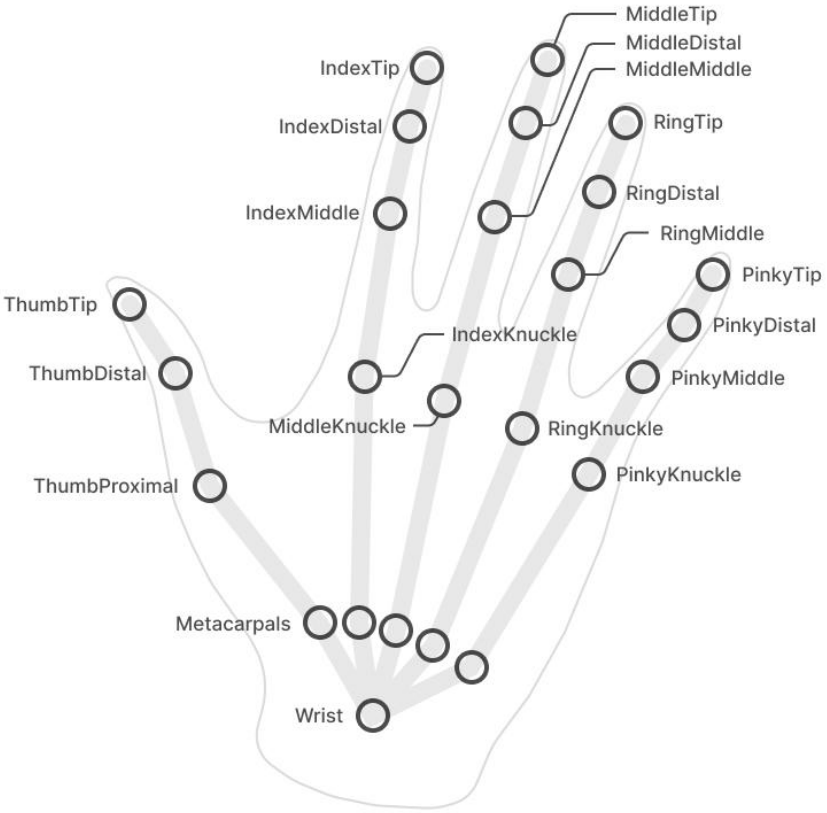
包括所有接口调用示例。

接口调用流程：

初始化 -> 注册回调监听 -> 开启相机，地图，平面检测 -> 处理回调数据 -> 暂停相机，地图，平面检测 -> 退出销毁。

3.8 手势功能介绍

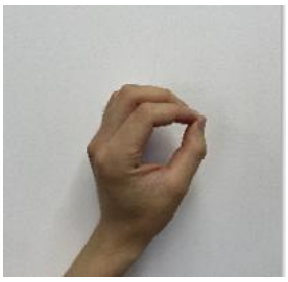
1.手势的 25 个特征点分布图:



2.静态手势码:

	ID	图解
UNKONOW	-1	
GRAB	0	

Point	1	
VICTORY	2	
Three	3	
Four	4	
OpenHand	5	
Call	6	

Rock	7	
Thumb	8	
Pinch	9	

3.手势接口

```
public static extern int start_gesture_ex(xslam_skeleton_callback cb,int type);
public delegate void xslam_skeleton_callback(XvXRSkeleton skeleton);
public struct XvXRSkeleton{
        public int size;
        public Point[] joints_ex;
        public RotatePoint[] rotateData;
        public int[] guestureID;
        public long dataFetchTimeMs;
        public long dataTimeStampMs;
};
```

四、unity 示例工程介绍

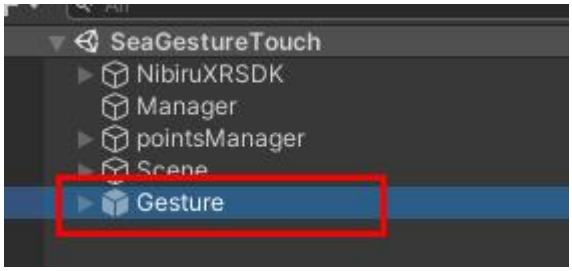
4.1 Deep Sea Demo

Demo 主要展示 slam 铆钉，3D 手势功能，APK 场景是一个大型沉浸式深海世界，并

通过手势触摸来对深海鱼类进行交互，场景中有多样的海底生物，包括鲨鱼、海豚等。通过手势可以触摸海底鱼类，在触摸后海底鱼类可以做出逃跑等动作。

详解：

下图是这个 Demo 场景中的基础手势组件，包含左右手的 25 个关节点位置信息。



1. 保存左右手 25 个手势关键点数据到 Transform 数组中，并启动手势识别功能

```
void start()
{
        for (int i = 0; i < 25; i++)
        {
            hand21List[i] = (lefthand21.GetChild(i).gameObject);
        }
```

```
            for (int i = 0; i < 25; i++)
            {
                hand21List[i + 25] = (righthand21.GetChild(i).gameObject);
            }
            GestreAction();
}
```

2. 开启手势识别功能，执行底层代码并执行回调函数，将数据在回调函数中进行处理

```
void GestreAction()
{
          Debug.Log("Gesture class start load ");
           try
           {
                AndroidJavaObjectjo=new
AndroidJavaObject("com.nibiru.ar.Gesture");
                float result;
                result = jo.Call<float>("getFloat", new object[] { });
                bool res = xslam_xrmoudle_open();
                start_gesture_ex(OnStartSkeletonCallback,1);
           }
           catch (AndroidJavaException e)
           {
                Debug.Log("Gesture class    load error ");
```

```
        }
}
```

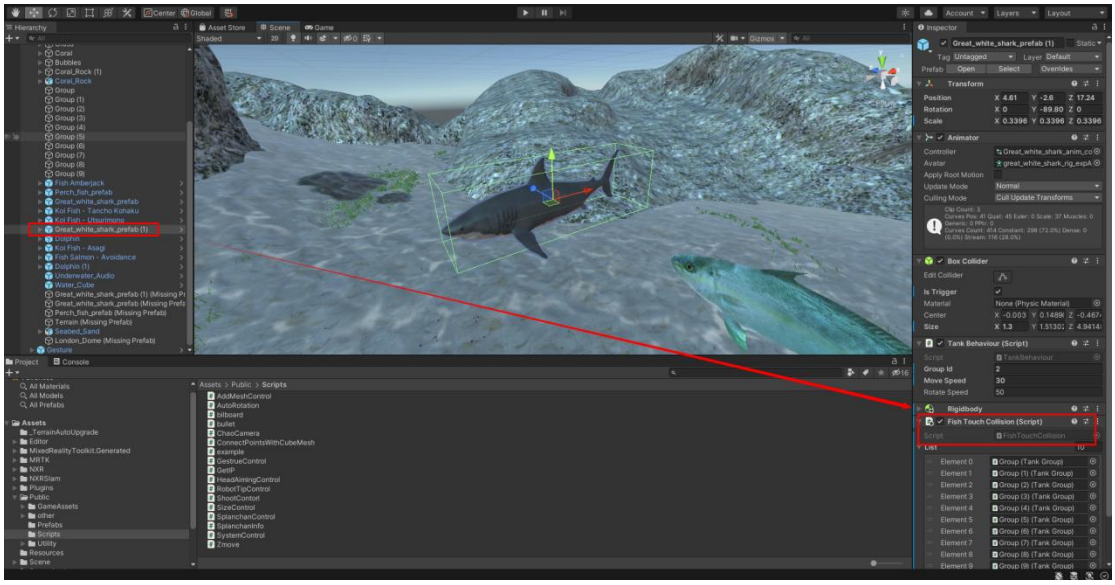
3. 手势数据回调函数分析

```
    [MonoPInvokeCallback(typeof(xslam_skeleton_callback))]
    public static void OnStartSkeletonCallback(XvXRSkeleton skeleton)
    {
        //位置
        for (int i = 0; i < joints_ex.Length; i++)
        {
            if (Double.NaN == skeleton.joints_ex[i].x || Double.NaN == skeleton.joints_ex[i].y || Double.NaN == skeleton.joints_ex[i].z)
            {
                joints_ex[i] = defultPoint;
            }
            else
            {
                if (skeleton.joints_ex[i].x == 0 && skeleton.joints_ex[i].y == 0 && skeleton.joints_ex[i].z == 0)
                {
                    joints_ex[i] = defultPoint;
                }
                else
```

```
                {
                    //获取返回的每个手势关节点位置，并保存在数组中
                    joints_ex[i] = new Vector3(skeleton.joints_ex[i].x,
-skeleton.joints_ex[i].y, skeleton.joints_ex[i].z);
                }
            }
        }
        //静态手势识别 ID 获取
        GestureLeftID = int.Parse(skeleton.guestureID[0].ToString());
        GestureRightID = int.Parse(skeleton.guestureID[1].ToString());
        //获取每个手势关节点的旋转角度
        for (int i = 0; i < rotation_ex.Length; i++)
        {
            RotatePoint rP = new RotatePoint();
            rP.x = skeleton.rotateData[i].x;
            rP.y = skeleton.rotateData[i].y;
            rP.z = skeleton.rotateData[i].z;
            rP.w = skeleton.rotateData[i].w;
            rotation_ex[i] = rP;
        }
    }
}
```

4. 更新左右手 25 个关节点位置，并显示静态手势 ID

```
void Update()
    {
        for (int i = 0; i < 50; i++)
        {
            if (double.IsNaN(joints_ex[i].x) == true)
            {
                continue;
            }
            hand21List[i].transform.position = joints_ex[i];
            Quaternion q = new Quaternion();
            q.x = -rotation_ex[i].x;
            q.y = rotation_ex[i].y;
            q.z = -rotation_ex[i].z;
            q.w = rotation_ex[i].w;
            hand21List[i].transform.rotation = q;
        }
        if (infoTxt != null)
            infoTxt.text = GestureLeftID + "===" + GestureRightID;
        if (idTxt != null)
            idTxt.text = GestureLeftID + "===" + GestureRightID;
}
```



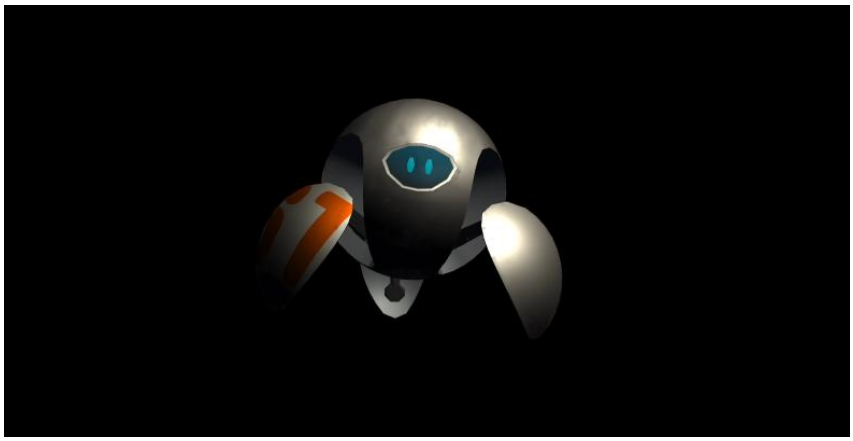
5. 鲨鱼的碰撞代码在 FishTouchCollision.cs 中

```
void OnTriggerEnter(Collider collider)
    {
        if (collider.name == "Cube (7)"){
list[this.gameObject.GetComponent<TankBehaviour>().groupId].transform.localPosition = new Vector3(Random.Range(-1,4), Random.Range(-1, -1), Random.Range(11, 15));
        }
}
```

检测到碰撞后鲨鱼会重新定位到新的终点位置。

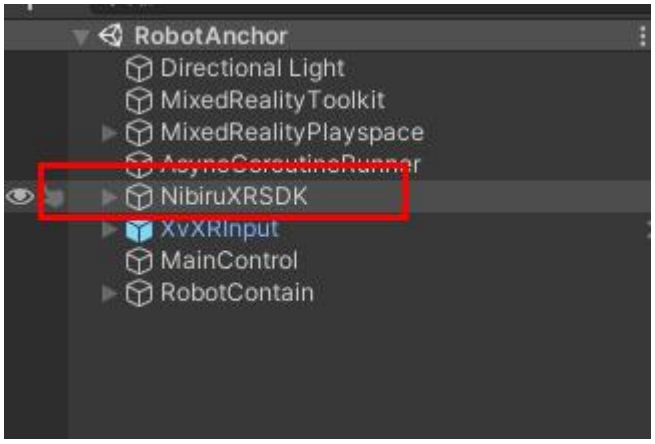
4.2 Robot Anchor Demo

Demo 主要展示 slam 空间锚定功能，场景中视角中间会预先锚定一个 robot 机器人。



可以通过 AR 眼镜看到 robot 锚定在现实场景中，同时通过 box 的 ok 键盘可以将 robot 显示在视角中心位置。

详解：



场景中的 NibiruXRSDK 就是场景中的头部摄像机位置，其中包括左右眼摄像机和头瞄点的设置。NibiruXRSDK 的位置需要放置在 unity 场景的(0,0,0)位置。

通过在 Update()里监听" Fire1"的 ButtonUp 事件来相应 Box 上 OK 按钮的点击操作

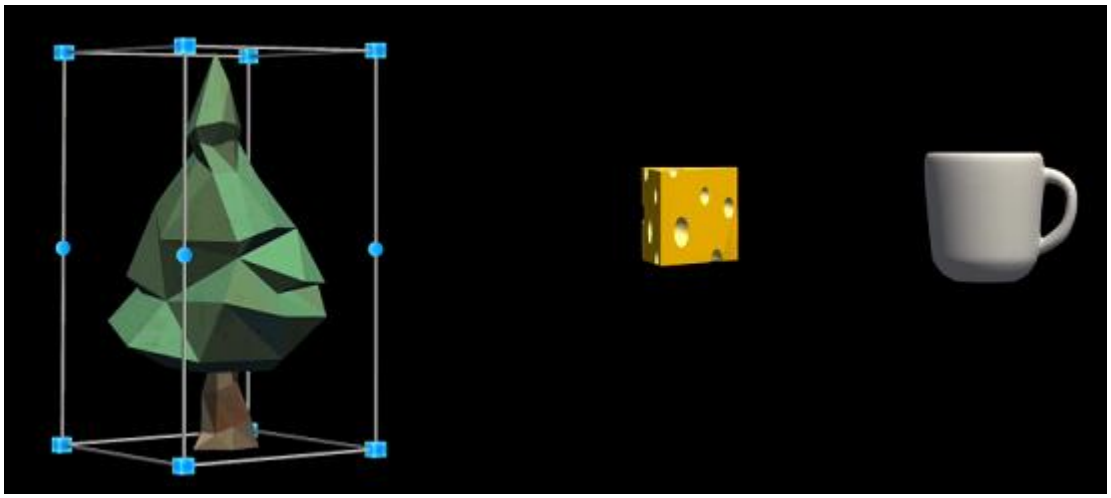
```
void Update()
    {
        //按下音量↑键生成并锚定物体
        if (Input.GetButtonUp("Fire1") || Input.GetKey(KeyCode.F1))
        {
            for (int i = 0; i < rootContain.transform.childCount; i++)
            {
                Destroy(rootContain.transform.GetChild(i).gameObject);
            }

            robot = GameObject.Instantiate(robotPrefab, robotPrefab.transform.position,
robotPrefab.transform.rotation, transform);
```

```
            robot.transform.parent = rootContain.transform;
            robot.transform.position = robotPrefab.transform.position + new
Vector3(0.069f, 0, 0);
            robot.SetActive(true);
        }
}
```

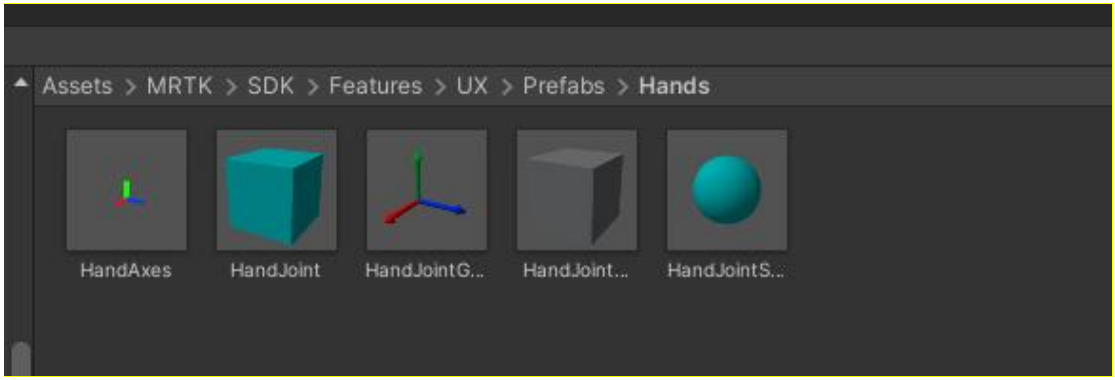
4.3 Gesture Demo

Demo 主要展示 3D 手势，SLAM，通过手势可以抓取和移动物体。



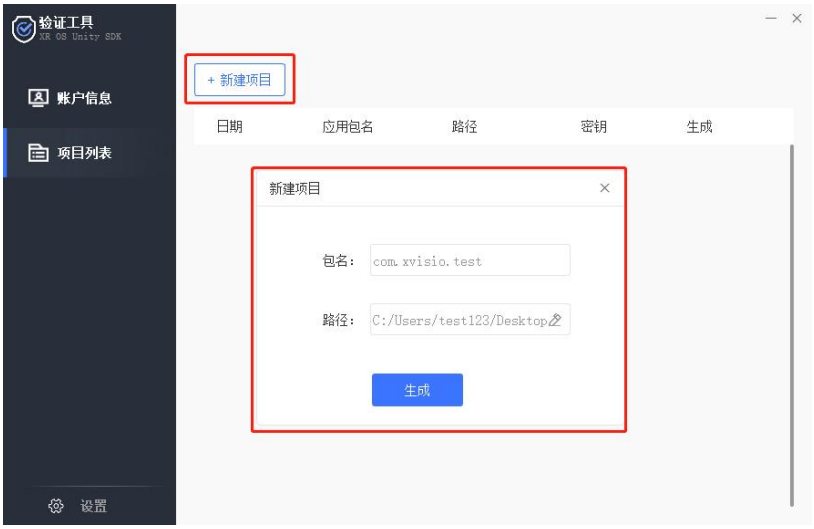
实例代码参考工程 Demo 场景 Gesture.unity，手势接口说明参考 3.8 章节。

替换手势关节点模型方法：



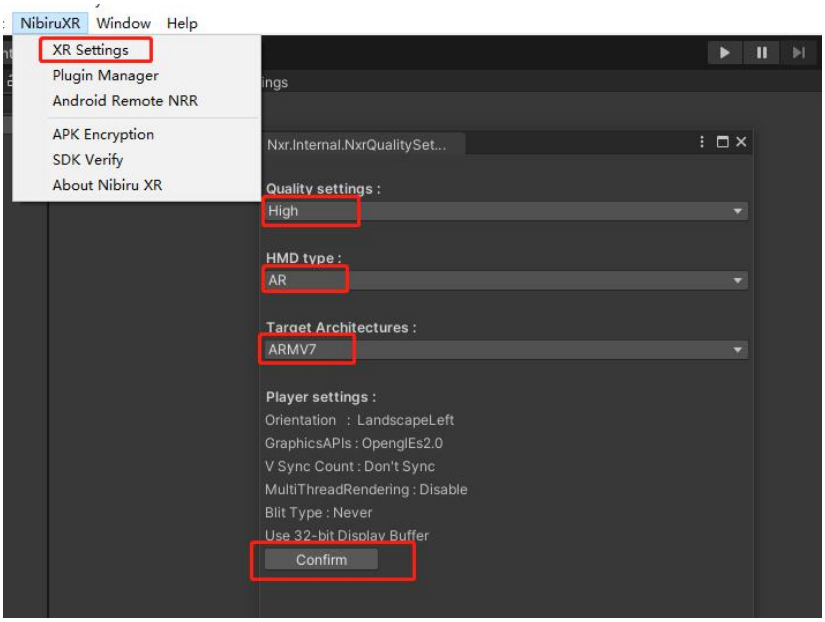
五、APK 打包步骤

1. 使用验证工具新建项目



包名与 unity PlayerSettings 中的 Package Name 保持一致
路径选择 unity 工程的 Assets\Plugins\Android\assets 目录
点击生成后，可以看见项目的密钥

2. Unity 发布前的设置
   点击 NibiruXR => XR Settings 设置相关参数后点击 Confirm 确认



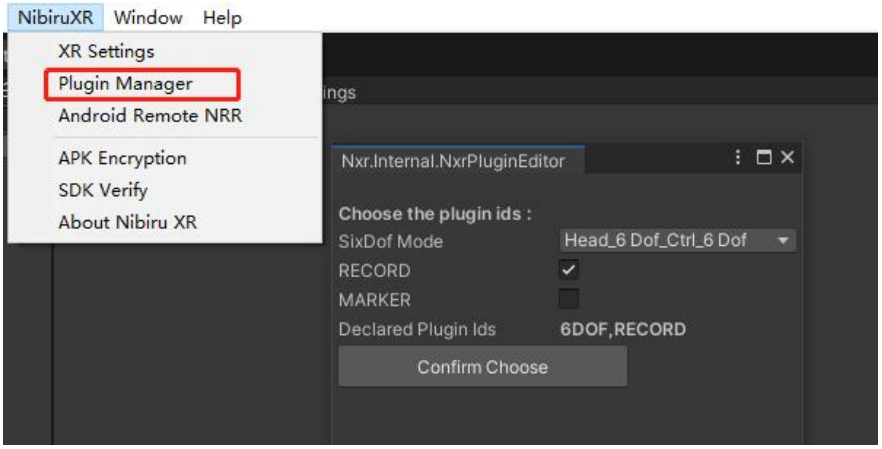
点击 NibiruXR => Plugin Manager 设置

SixDof Mode：6dof 模式，根据需求选择头控 6dof 或 3dof 模式

RECORD：是否启用 RECORD 功能

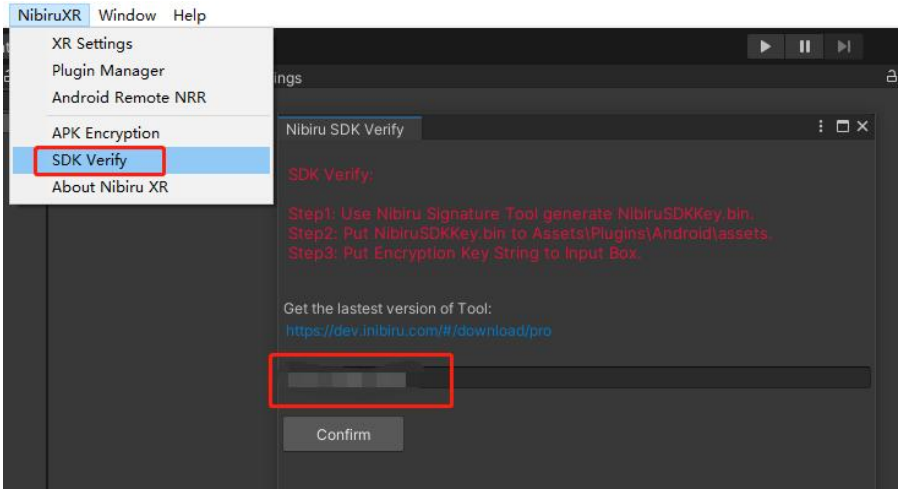
MARKER：是否启用 MARKER 功能

点击 Confirm Choose 完成设置



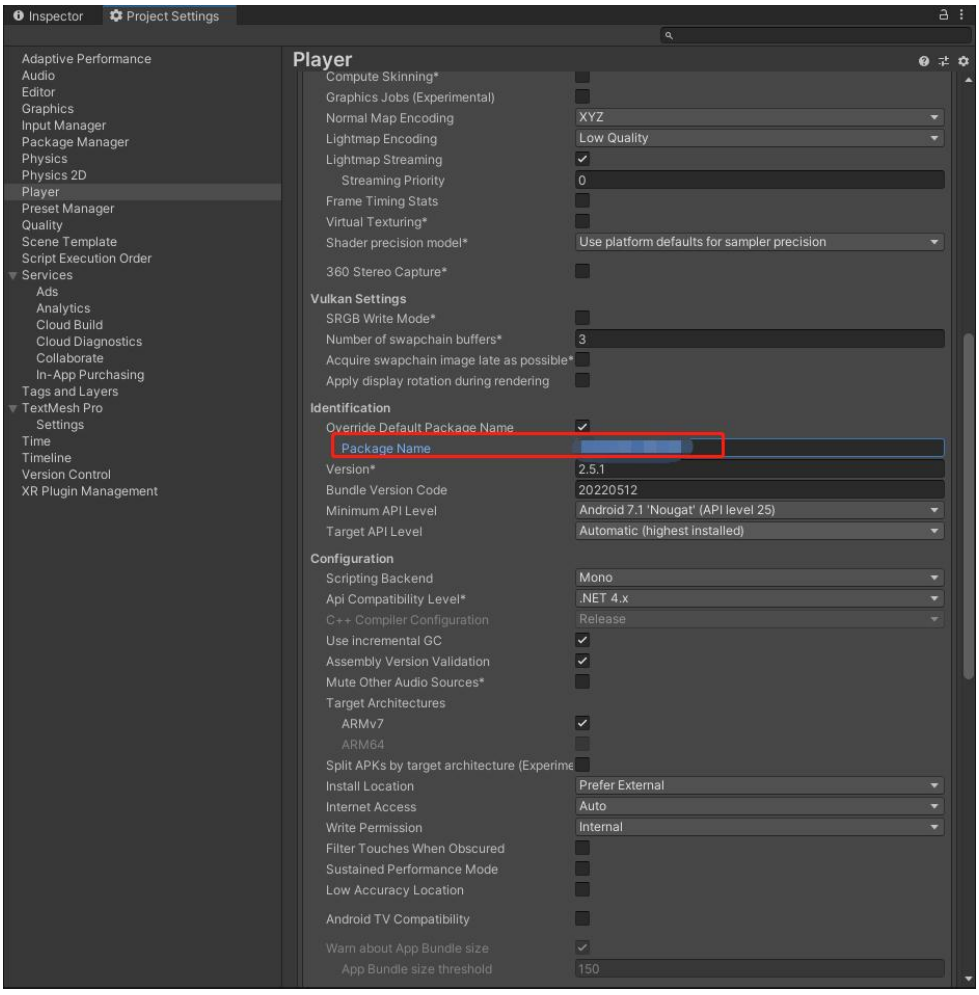
点击 NibiruXR => SDK Verify 设置密钥

填写验证工具生成的密钥



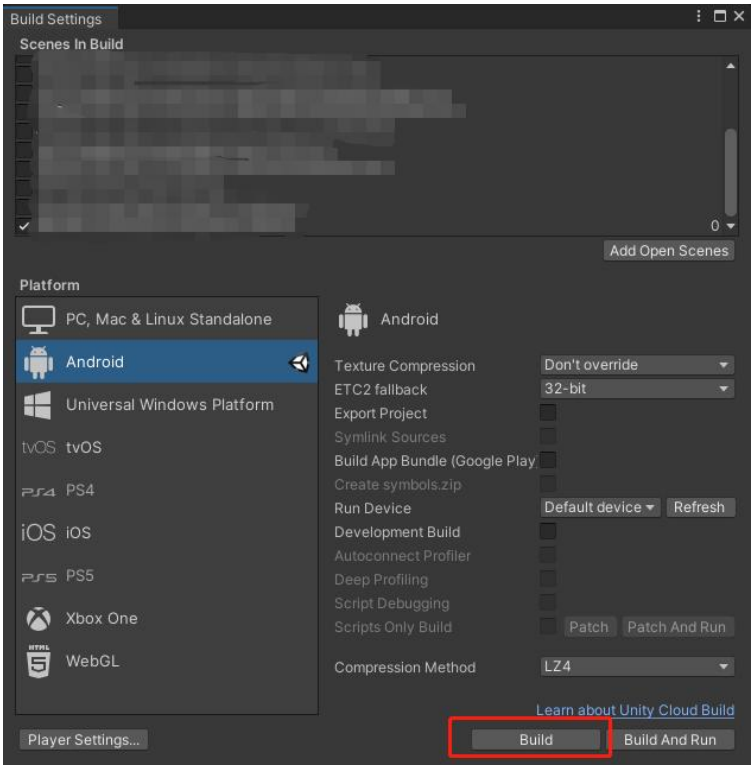
设置 PlayerSettings

Other Settings 中的 Package Name 须与验证工具中的包名保持一致



3. 打包发布 apk

   勾选要发布的场景，点击 Build 发布 apk



六、内容开发标准规范

1) 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。

2) 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）。

3) 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。

4) 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

5) 尽量减少或不使用 Unity 灯光，采用 Lightingmap 烘培光影信息。

6) 尽量减少或不使用粒子系统。

七、FAQ

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。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?

检查 AndroidManifest 是否加上了 NVR 标签。

检查脚本挂载的相机是不是 MainCamera。

3、为什么在打包 APK 时失败?

检查 Android 引用的 jar 包是否有冲突。

在编辑器查看 Console 错误日志，具体分析原因。

4、为什么打包成功，但是运行崩溃或黑屏?

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。

: 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。

检查 Plugins/Android 目录下的 jar 是否有重复。

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。

5、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