秘别： 编号：

Skyworth Standalone VR Video SDK（Unity）

开发文档

拟制 _____日期 _____

审核 _____日期 _____

批准 _____日期 _____

深圳创维新世界科技有限公司

1 简介

1.1 SDK 介绍

SDK 支持硬件设备：创维 VR 一体机 S8000。

SDK 主要提供：3dof 手柄/Nolo 手柄/锚点交互支持，多功能交互支持，双目立体渲染等功能。

1.2 开发环境

Unity：目前支持 Unity 2017.2.0 及以上版本，推荐使用 Unity2017. 4.6f1 版本。

Android SDK：API Level 19 及以上，推荐使用 API Level 25。

JDK： jdk1.7.0_01 及以上。

1.3 SDK 构成

为方便开发者使用，SDK 通过 UnityPackage 包的形式提供，开发者导入后可见如下目录：

图 1.1 SDK 结构

1.4 SDK 说明

Svr-Video-Unity-SDK 需要配合 SVR-Unity-SDK 使用，请首先按照“Skyworth Standalone VR SDK-Unity-SDK_Developers_Guide_xx_xxxx”文档配置 SVR-Unity-SDK 后再导入 Svr-Video-Unity-SDK。导入 SDK 之后若出现如下图 1.2 错误提示，请参照“2.4 项目设置”配置开发项目。

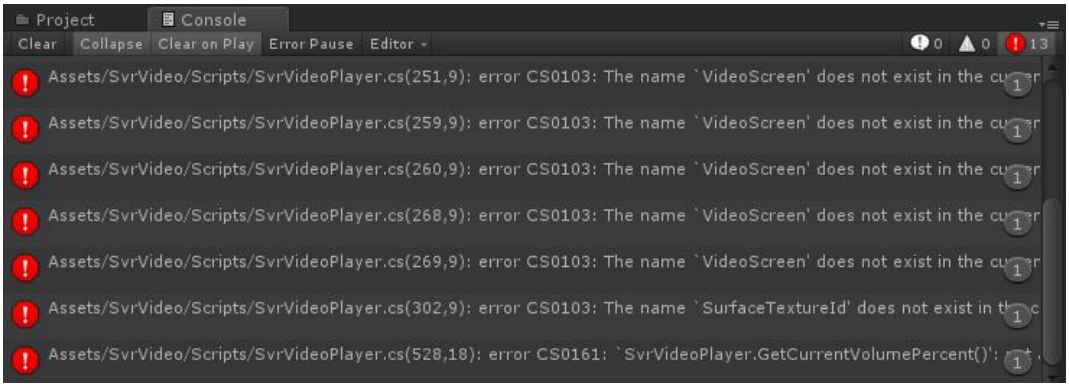

图 1.2 错误提示

2 SDK 的使用说明

2.1 新建工程

打开 Unity，新建工程如下：

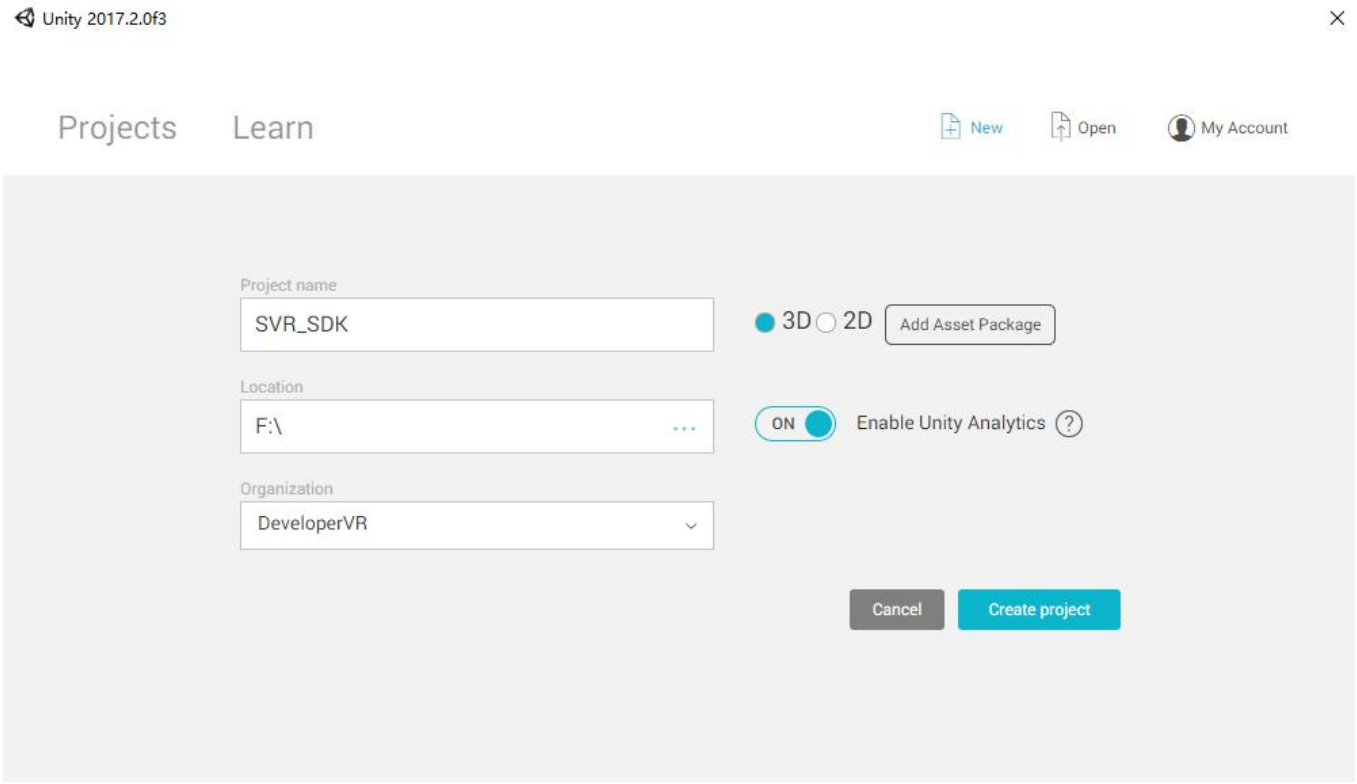

图 2.1 新建工程

2.2 导入 SDK

选择菜单 Assets->Import Package->Custom Package…如下：

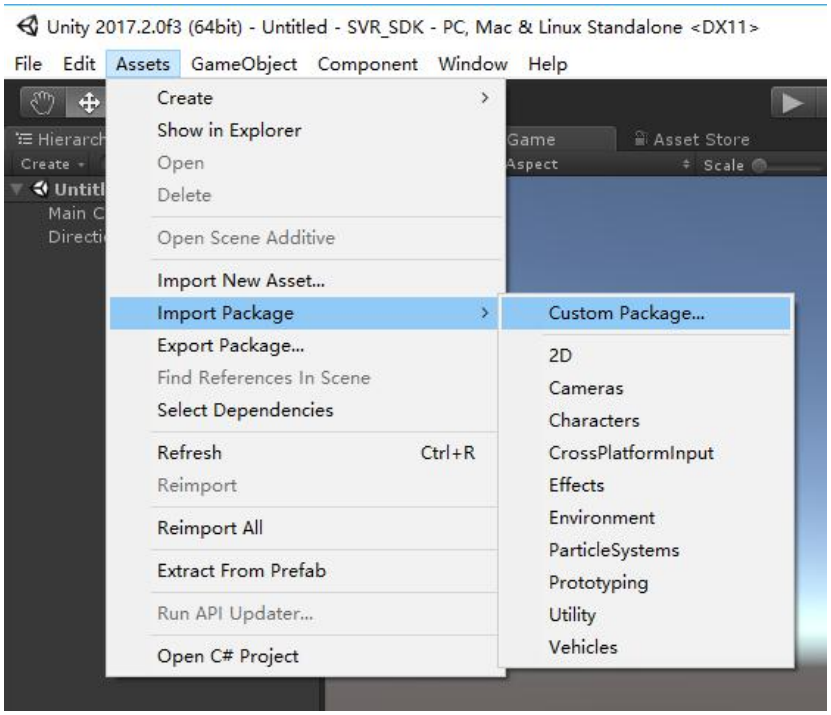

图 2.2 导入 SDK

此时 Unity 会弹出文件选择对话框，选择“Svr-Video-Unity-SDK-1.x.x.unitypackage”后点击打开如下：

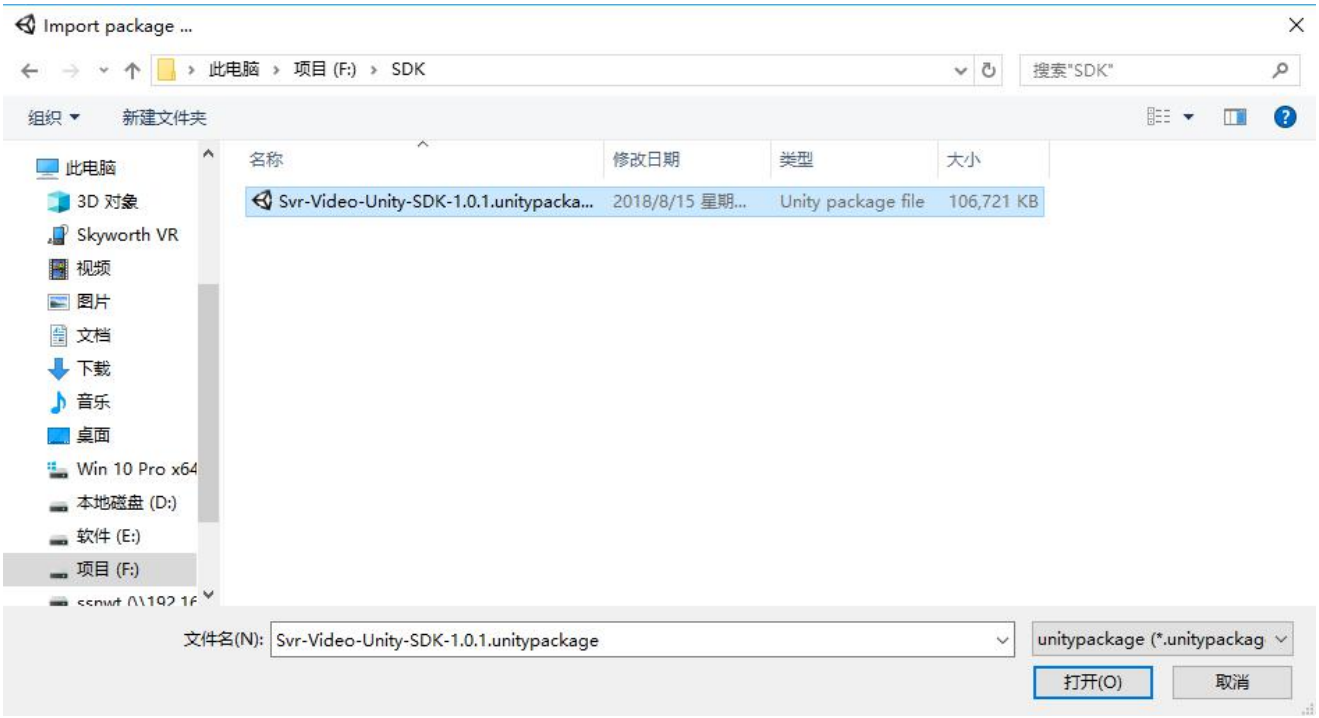

图 2.3 选择 Unity Package

点击后返回至 Unity 界面，系统弹出 SDK 目录层级，请根据需要导入：

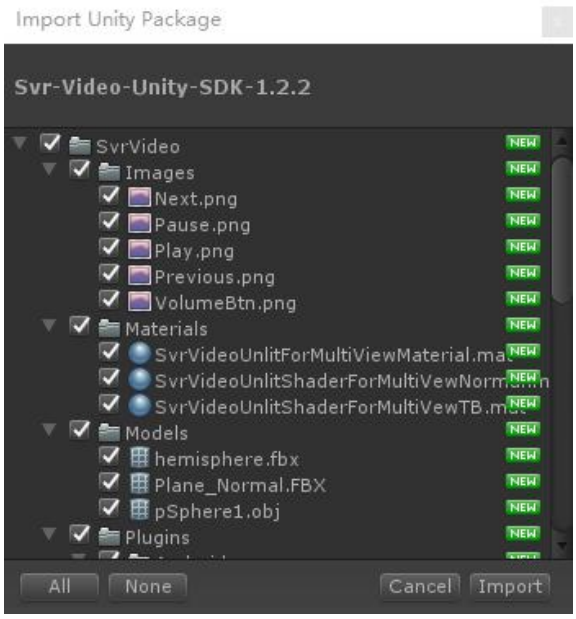

图 2.4 导入选项

2.3 使用 SDK

进入 Project 选项卡，依次展开 Assets->SvrVideo->Scenes，选择 SkyworthVRVideoDemo 场景，点击运行按钮，在 Game 窗口中可看到如下：

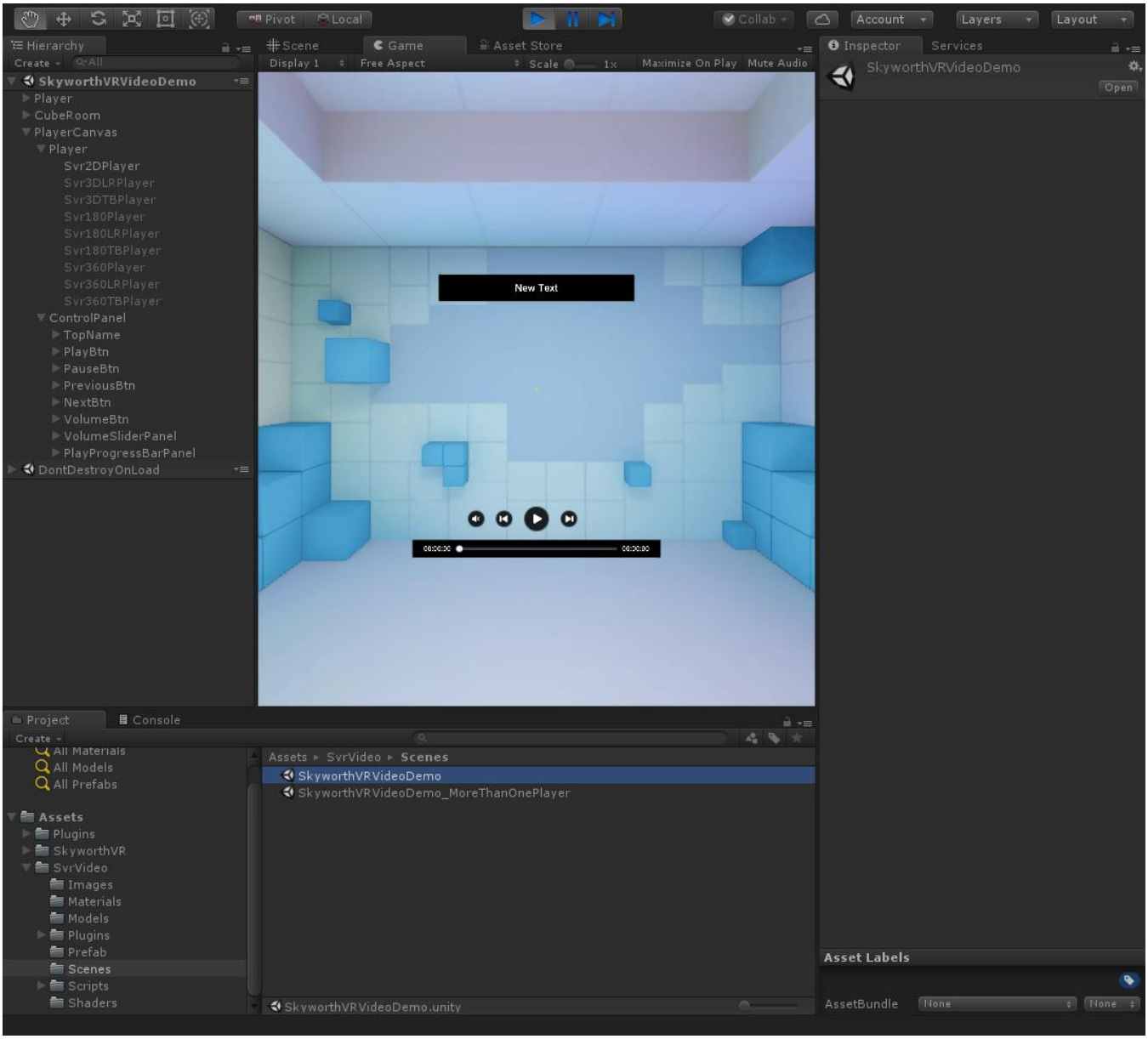

图 2.5 模拟运行

按住 Alt 移动鼠标，可上下左右转动画面；按住 Ctrl 移动鼠标，可上下翻转画面；按住 Shift 移动鼠标，可模拟手柄操作。

2.4 项目设置

2.4.1 AndroidManifest.xml 配置

如图 2.6 中所示，AndroidManifest 中在<application>标签中加入“android:largeHeap="true"”字段，以防止播放视频时运行内存占用过大造成内存泄漏。AndroidManifest 配置如下图 2.6 所示：

图 2.6 AndroidManifest 配置示意图

2.4.2 QualitySettings 设置

如图 2.7 中 Levels 选择安卓平台绿色勾选项的“Medium”等级。具体参数如下图 2.7 所示：

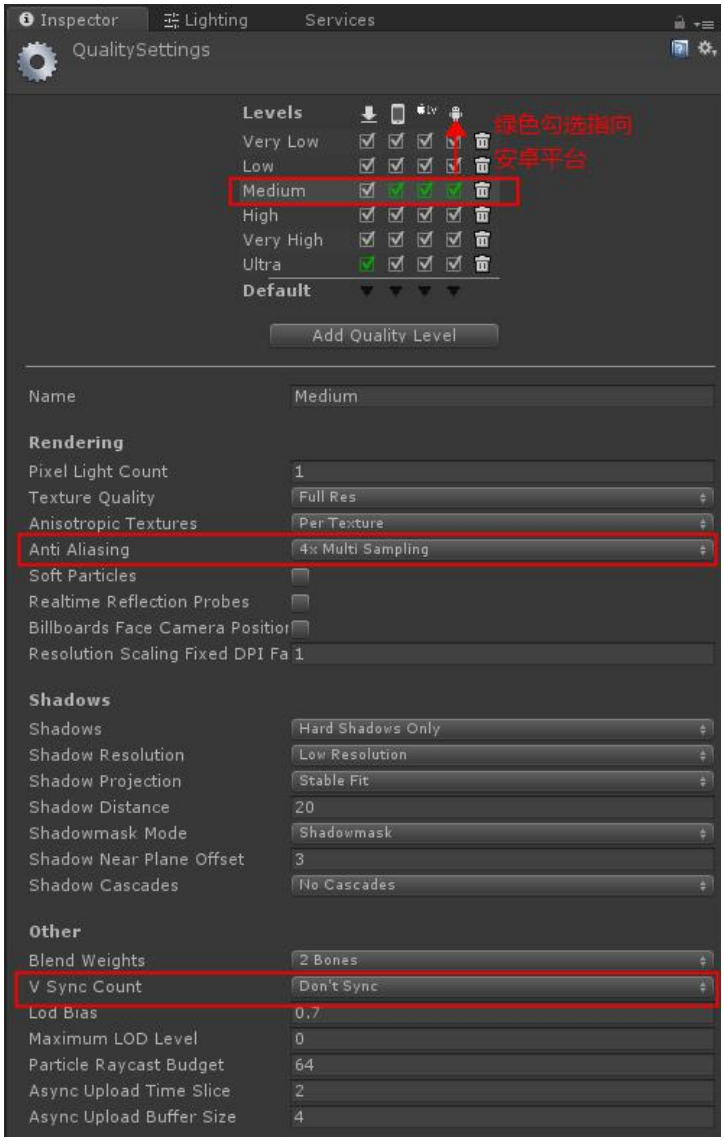

图 2.7 QualitySettings 设置示意图

Anti Aliasing：可以根据需要进行调整，推荐使用 4 倍抗锯齿。

V Sync Count：一定要设置为 Don’t Sync。

注意：以防止运行的异常，建议开发者将所有等级都如上图配置。

2.4.3 PlayerSettings 设置

1. Resolution and Presentation 选项卡中，导出设置为横屏。具体参数如下图 2.8 所示：

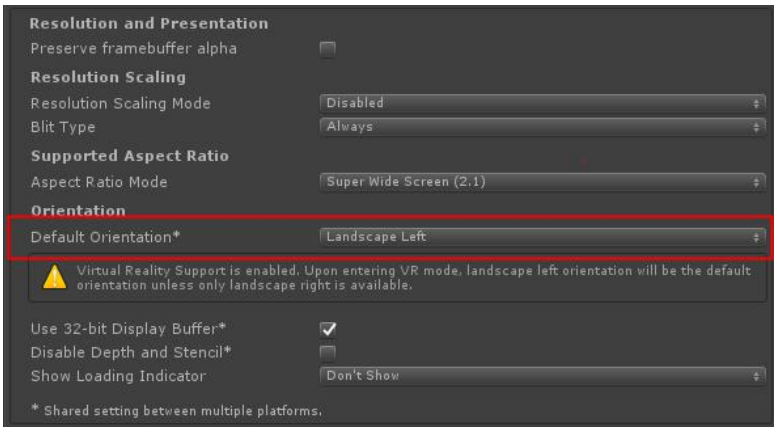

图 2.8 导出设置示意图

2. Other Settings 选项卡中，不要勾选 GPU Skinning，API Level 为 Android 7.1，读写权限根据需要选择 SDCard。具体参数如下图 2.9 所示：

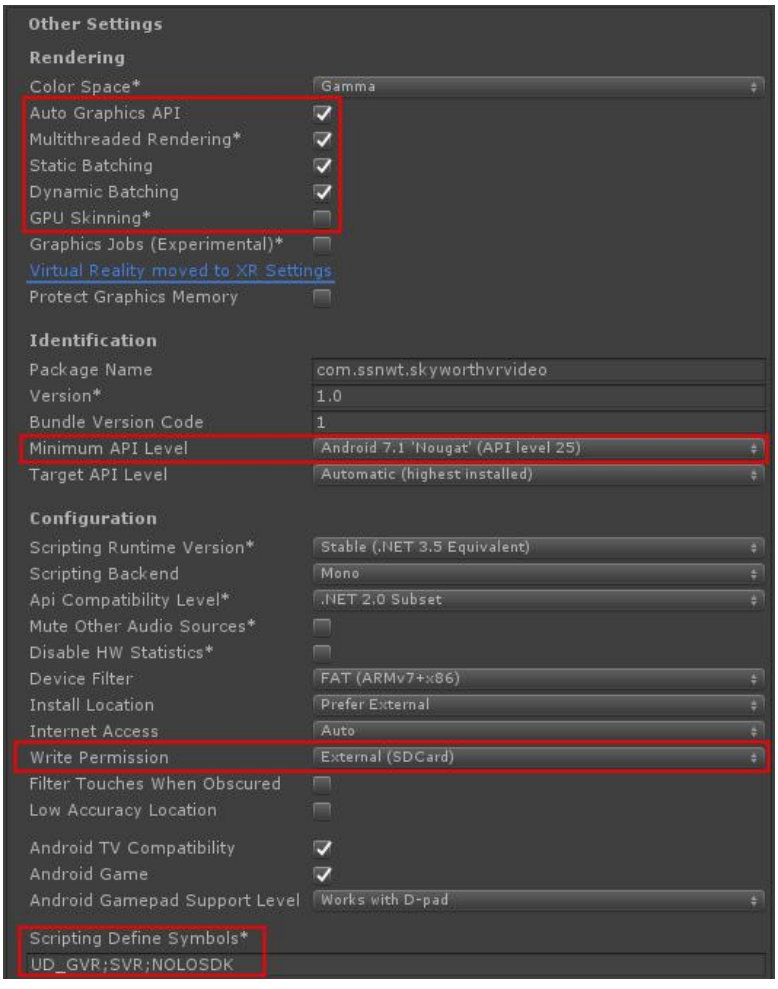

图 2.9 PlayerSettings 设置示意图

3. XR Settings 选项卡中，勾选 Virtual Reality Supported，选择“Mock HMD - Vive”，将 Stereo Rendering Methods 设置为“Single Pass(Preview)”。具体参数如下图 2.10 所示：

图 2.10 XR Settings 设置示意图

2.4.4 Build Settings 设置

选择默认平台“Android”，构建系统(Build System)选择“Internal”编译方式。具体参数如下图 2.11 所示：

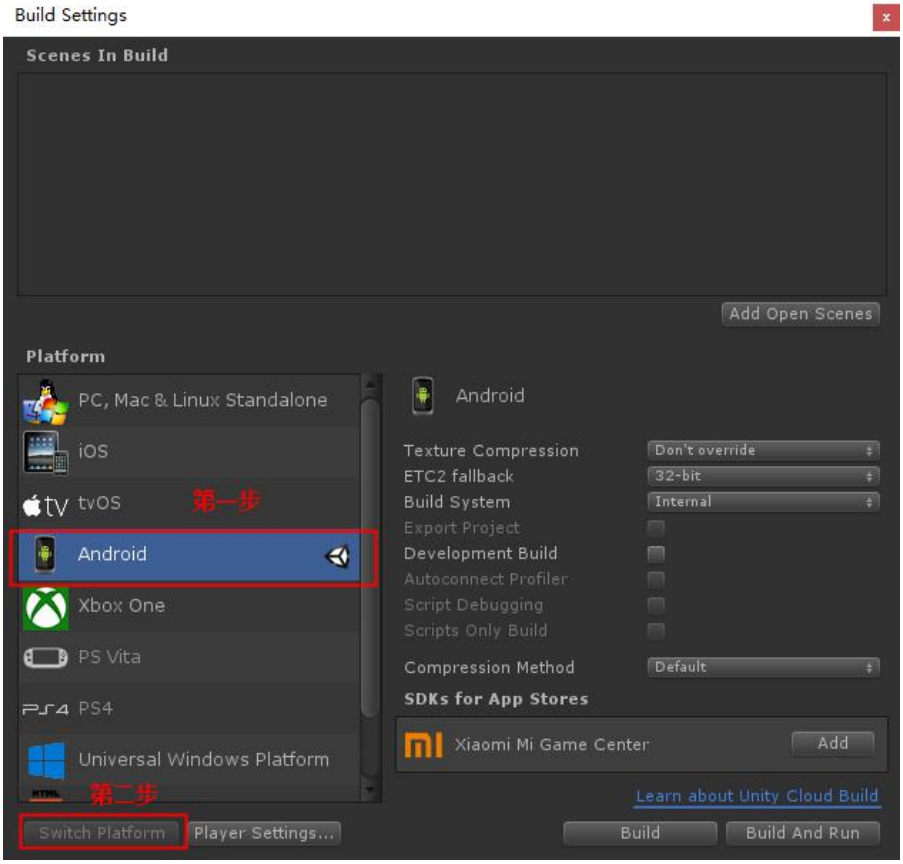

图 2.11 Build Settings 设置示意图

2.5 导出到设备运行

1.将设备通过 USB 连接到电脑，当连上电脑之后，Launcher 界面的电池图标会变成充电状态。

2.点击 File->Build & Run，等待进度条结束就 OK 了！

3 SvrVideo 使用指南

3.1 SvrVideo 说明

该 SDK 提供单一播放器、多个播放器两种播放模式。单一播放器 Demo 中提供平面视频、180 度视频、360 度视频的功能，可以支持普通 2D、3D 上下格式、3D 左右格式的视频播放。在 SDK 里提供了视频格式、播放、暂停、跳转进度、循环播放、调整音量、获得视频宽高度等功能。多个播放器 Demo 中主要展示多个普通 2D 播放器同时播放的功能。

3.2 SvrVideo 介绍

3.2.1 单一播放器 Demo

该 Demo 场景在 Assets->SvrVideo->Scenes->SkyworthVRVideoDemo。Demo 中有 3 个主要的对象，如下图 3.1 所示：

图 3.1 SkyworthVRVideoDemo 场景

Player：控制摄像机，手柄或凝视锚点，Event 交互，头部追踪，是每个场景必不可少的对象，详见第五节“GoogleVR API 接口函数”。Player 对象有两种：如果需要支持 Nolo 手柄，可以在 Assets->SkyworthVR->NoloVR->Prefabs 中获得，默认支持 Nolo 手柄；如果不需要支持 Nolo 手柄，可以在 Assets->SkyworthVR->GoogleVR->Prefabs 中获得。

PlayerCanvas：控制播放器（Player），控制面板（ControlPanel）

CubRoom：场景对象

3.2.2 多个播放器 Demo

该 Demo 场景在 Assets->SvrVideo->Scenes->SkyworthVRVideoDemo_MoreThanOnePlayer。Demo 中有 3 个主要的对象，如下图 3.2 所示：

图 3.2 SkyworthVRVideoDemo_MoreThanOnePlayer 场景

Player：控制摄像机，手柄或凝视锚点，Event 交互，头部追踪，是每个场景必不可少的对象，详见第五节“GoogleVR API 接口函数”。

PlayerCanvas：控制播放器（Player）

CubRoom：场景对象

3.3 播放器说明

Svr 播放器提供普通 2D、3D 上下格式、3D 左右格式的平面视频、180 度视频、360 度视频的功能。播放器预制体可以在 Assets->SvrVideo->Prefab 中获得。播放器示意如下图 3.3 所示：

图 3.3 播放器对象示意图

3.3.1 立体格式说明

1. 立体格式分为普通 2D、3D 上下格式、3D 左右格式，现以平面视频为例。平面视频提供三种播放器，其播放器对象均使用同一个 3D 平面模型，如下图 3.4 所示：

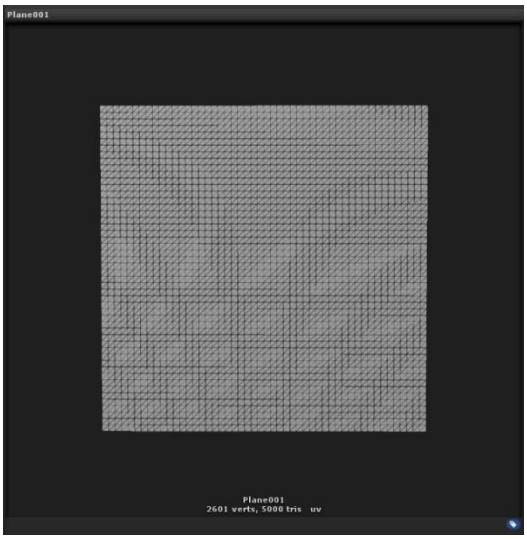

图 3.4 3D 平面模型

2. 无论立体格式是什么，只需要一个播放器对象即可。

3. 播放器对象需要绑定播放器控制脚本“SvrVideoPlayer”，该脚本中提供所有的播放器控制接口，如下图 3.5 所示：

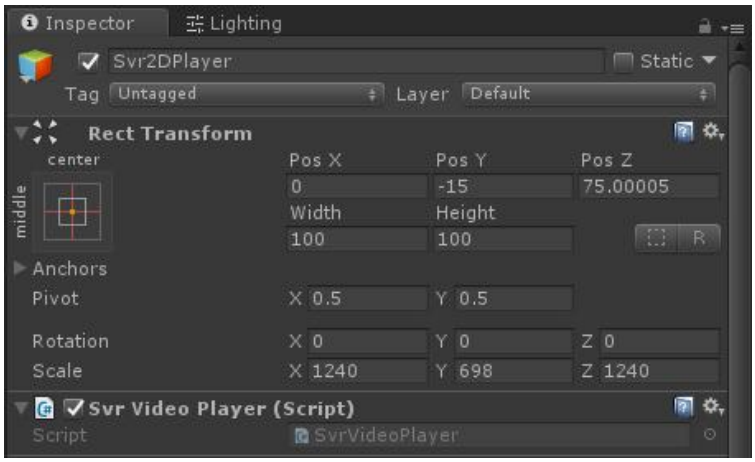

图 3.5 播放器对象绑定 SvrVideoPlayer 脚本

4. 播放器对象需要绑定着色器脚本“Svr/SvrVideoUnlitShaderForMultiView”，如下图 3.6 所示：

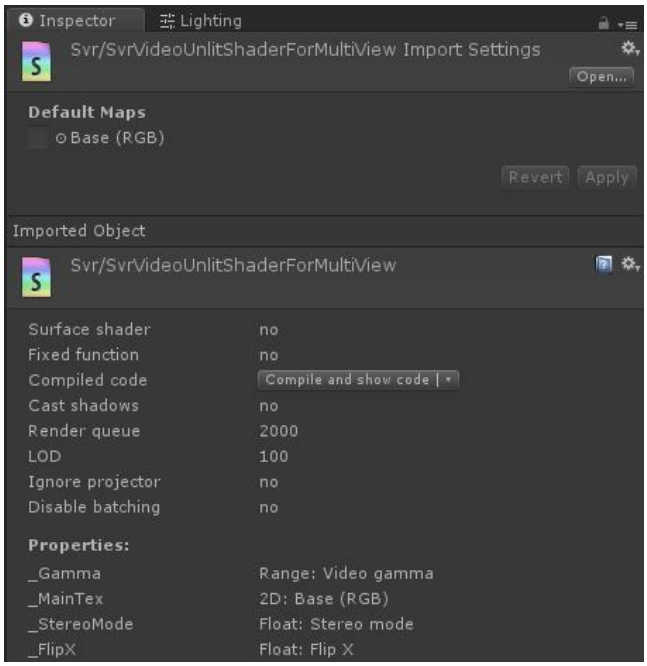

图 3.6 Shader 脚本

该 Shader 脚本控制图像的渲染，其中有两个预处理指令：_STEREOMODE_TOPBOTTOM、_STEREOMODE_LEFTRIGHT，这两个预处理指令可以控制左右眼摄像机渲染的图像的像素取值，可以实现普通 2D、3D 上下格式、3D 左右格式的视频播放。开发者不需要设置或修改该 Shader 脚本，SDK 为开发者提供格式变换的方法，只需要调用此方法即可，详见第 4 节。

3.3.2 视频模式说明

180 度视频、360 度视频模式与平面视频相同，均使用播放器控制脚本“SvrVideoPlayer”与着色器脚本“Svr/SvrVideoUnlitShaderForMultiView”，需要替换播放器对象的 3D 模型，根据需要修改播放器的大小规模，SDK 场景中为开发者提供平面视频、180 度视频、360 度视频的三种模型实例，以作参考。

1. 180 度视频使用半球形模型，如下图 3.7 所示：

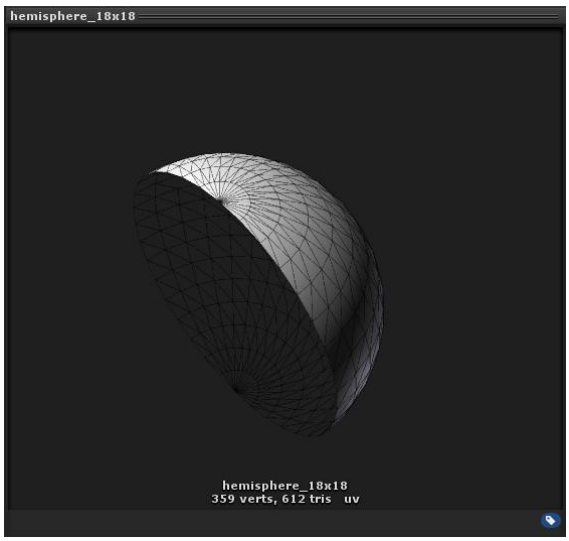

图 3.7 180 度视频模型

2. 360 度视频使用球体模型，如下图 3.8 所示：

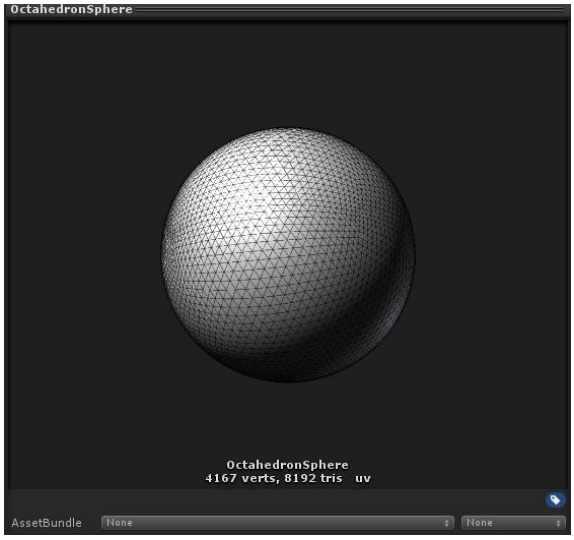

图 3.8 360 度视频模型

3.3.3 视频模式切换说明

SDK 提供五种播放器示例，详情参照“图 3.3 播放器对象”。

1. 平面视频有三种，分别为：Svr2DPlayer、Svr3DLRPlayer、Svr3DTBPlayer。

2. 三种平面视频的模型为同一平面模型，只需要修改 Stereo Type 即可，2D 视频选择“None”，左右 3D 视频选择“Left Right”，上下 3D 视频选择“Top Bottom”。详情参照“3.3.4 SvrVideoPlayerDemo 说明”。

3. 平面视频、180 度视频与 360 度视频区别只在于模型不同，模型详情参照“3.3.2 视频模式说明”。

4. 切换不同的播放器模式时，在 Inspector 面板中选择需要的对象，然后修改 SvrVideoPlayerDemo 中“Video Urls”视频地址选项，如下图 3.9：

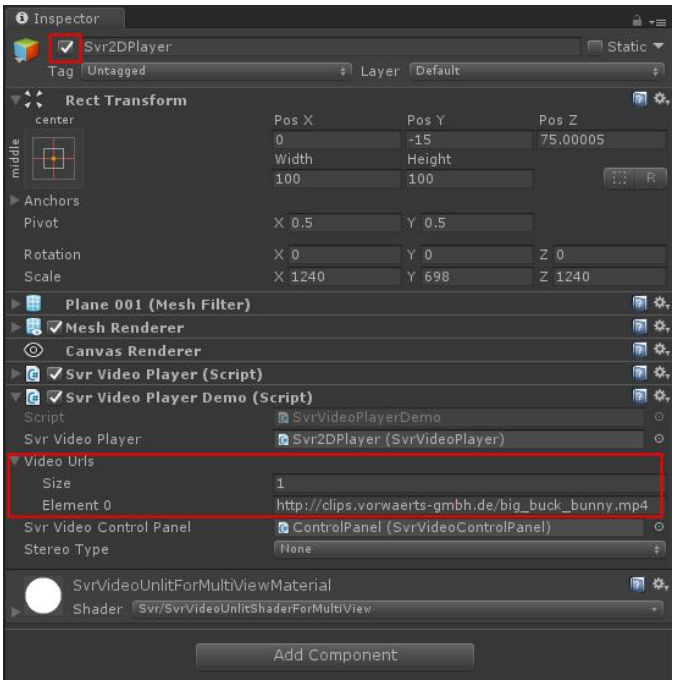

图 3.9 切换播放器

5. 更换视频模式之后，需要重新关联播放器 UI 显示控制对象，详情参照“3.4 SvrVideoControlPanel 说明”。

3.3.4 SvrVideoPlayerDemo 说明

此脚本是播放器调用脚本，控制视频的播放与控制面板的属性设置看，获得回调函数。将此脚本绑定到场景中的对象上，在 Inspector 面板中设置成员变量的值。

1. 需要设置 video 的播放器对象、播放地址、关联绑定 ControlPanel、立体类型，立体类型可以根据不同的视频选择相应的立体格式，也可以从代码中控制。如下图 3.10：

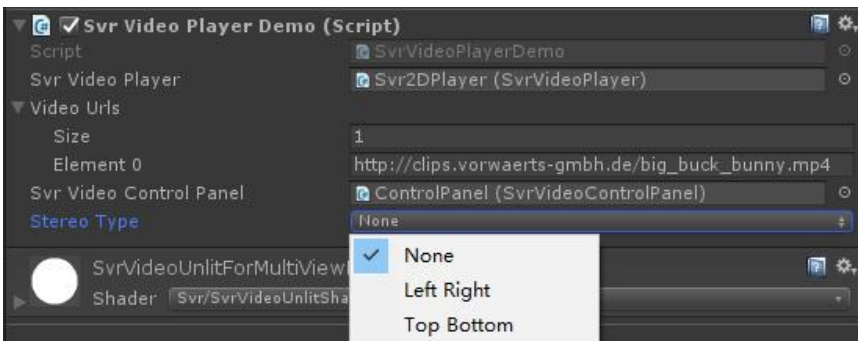

图 3.10 SvrVideoPlayerDemo 配置

2. 绑定 SvrVideoPlayer 播放器脚本的回调

```
SvrVideoPlayer.OnEnd += OnEnd;
SvrVideoPlayer.OnReady += OnReady;
SvrVideoPlayer.OnVolumeChange += OnVolumeChange;
SvrVideoPlayer.OnProgressChange += OnProgressChange;
SvrVideoPlayer.OnVideoError += OnVideoError;
```

3. 播放视频之前必须先设置视频数据，初始化播放器，不然播放会失败

```
// Set video data source
SvrVideoPlayer.PreparedPlayVideo(VideoUrls[currentPlayingVideoIndex]);
```

4. 根据“图 3.8 SvrVideoPlayerDemo 配置”中所选择的立体格式类型设置 Shader 的渲染属性

```
// Set video stereo mode
if (StereoType == StereoType.LeftRight)
   SvrVideoPlayer.SetPlayMode3DLeftRight();
else if (StereoType == StereoType.TopBottom)
   SvrVideoPlayer.SetPlayMode3DTopBottom();
else
   SvrVideoPlayer.SetPlayMode2D();
```

5. 设置控制面板中显示的视频名字等信息

```
SvrVideoControlPanel.SetVideoName(name);
SvrVideoControlPanel.SetPlayControlButtonStatus(true);
SvrVideoControlPanel.SetVideoCurrentTime(0);
```

3.4 SvrVideoControlPanel 说明

此脚本是播放器 UI 显示控制，控制当前播放器的显示信息包括视频名字、音量调节、上一首视频、下一首视频、播放/暂停、播放进度条显示、当前播放时间节点、视频总时长。该预制体可以在 Assets->SvrVideo->Prefab 中获得，如下图 3.11 所示：

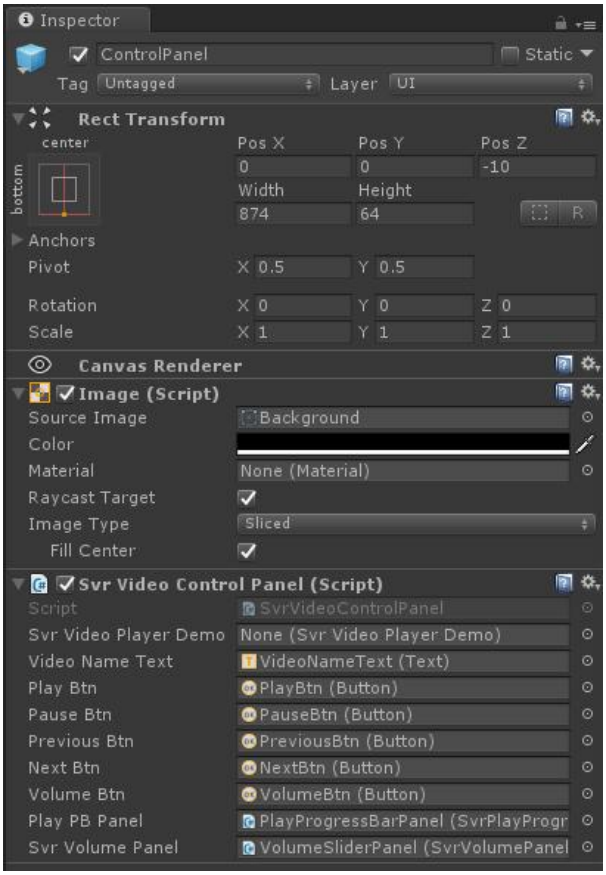

图 3.11 SvrVideoControlPanel 配置

3.4.1 单一 ControlPanel 说明

该 Demo 中全局使用同一个 ControlPanel 对象，切换不同播放器时，需要将“SVR Video Player Demo”重新关联。根据需要更换播放器，如下图 3.12 所示：

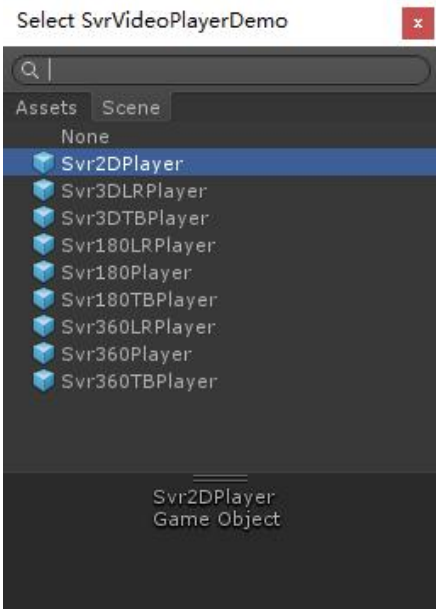

图 3.12 选择 SVR Video Player Demo

注意：SvrVideoControlPanel 中 UI 面板显示信息由 SvrVideoPlayerDemo 控制，详情参见“3.3.4

SvrVideoPlayerDemo 说明”。也可以为每个播放器对象绑定一个 ControlPanel，详情参见“3.4.2 多个 ControlPanel 说明”。

3.4.2 多个 ControlPanel 说明

该 Demo 中每个播放器对象下均创建单独的 ControlPanel 对象。将 ControlPanel 与父类对象播放器相互关联即可控制当前播放器的 UI 显示，如下图 3.13 所示：

图 3.13 多个 ControlPanel

4 SvrVideo API 接口函数

4.1 SetPlayMode2D 说明

功能：

设置播放视频的立体格式为普通 2D 视频。

使用：

在视频播放时使用 SvrVideoPlayer 的实例化对象调用即可，可以随时切换立体格式。

4.2 SetPlayMode3DLeftRight 说明

功能：

设置播放视频的立体格式为普通左右格式视频。

使用：

在视频播放时使用 SvrVideoPlayer 的实例化对象调用即可，可以随时切换立体格式。

4.3 SetPlayMode3DTopBottom 说明

功能：

设置播放视频的立体格式为上下格式视频。

使用：

在视频播放时使用 SvrVideoPlayer 的实例化对象调用即可，可以随时切换立体格式。

4.4 CreatVideoPlayer 说明

功能：

创建播放器，绑定播放器事件，绑定底层播放器播放事件回调方法，创建图像矩阵。

使用：

需要播放视频之前使用 SvrVideoPlayer 的实例化对象调用。可以在任意时刻多次创建播放器。

注意：

1.必须在调用 PreparedPlayVideo 之前使用，CreatVideoPlayer 需要对应调用 Release；

2.只创建一次播放器，多次 PreparedPlayVideo。只创建一次播放器，只需调用一次 Release，即播放视频之前创建播放器 CreatVideoPlayer->初始化播放器 PreparedPlayVideo->播放完成 Stop->初始化播放器 PreparedPlayVideo->播放完成 Stop->...->销毁播放器 Release；

3.允许多次创建播放器，即每次播放视频时创建播放器 CreatVideoPlayer->初始化播放器 PreparedPlayVideo->播放完成 Stop->销毁播放器 Release->播放另一部视频 CreatVideoPlayer->初始化播放器 PreparedPlayVideo->...。

4.5 Release 说明

功能：

销毁播放器。

使用：

不需要播放器时使用 SvrVideoPlayer 的实例化对象调用。

4.6 PreparedPlayVideo 说明

功能：

配置播放器所需的数据，初始化播放器，每次播放视频均需要调用。

使用：

在视频播放之前调用，需要传入视频地址。设置数据需要创建 Json 字符串，此 Json 字符串是由 JVideoDescriptionInfo 类转换而成的 Json 数据，再传入底层为播放器配置播放数据。

注意：

在调用之前必须已经调用过 CreatVideoPlayer。

4.7 Play 说明

功能：

播放视频。

使用：

在视频未播放时，播放视频，并且需要视频已经准备好之后在调用。

4.8 Pause 说明

功能：

暂停视频。

使用：

在视频播放时，暂停视频。

4.9 Stop 说明

功能：

停止视频，每次停止播放需要调用。

使用：

在视频播放、暂停、结束状态下，停止视频，重置播放器（ResetPlayer），释放图片资源。

4.10 SeekToTime 说明

功能：

跳转播放进度。

参数：

需要跳转到的进度位置，类型 long，单位毫秒。

4.11 SetLoop 说明

功能：

设置是否循环播放当前视频，默认不循环，播放完成后暂停播放。

参数：

是否循环播放，类型 bool，默认为 false。

使用：

需要在视频准备完毕之后，正在播放时调用。

4.12 SetCurrentVolumePercent 说明

功能：

调节设备音量百分比。

参数：

音量百分比，类型 float，取值范围 0~1。

4.13 GetMaxVolume 说明

功能：

获得设备的最大音量值。

返回值：

音量最大值，类型 int，取值范围 0~15，0 为静音。

4.14 GetCurrentVolume 说明

功能：

获得设备当前的音量值。

返回值：

音量当前值，类型 int，取值范围 0~15，0 为静音。

4.15 GetCurrentVolumePercent 说明

功能：

获得设备音量百分比。

返回值：

音量百分比，类型 float，取值范围 0~1。

4.16 GetVideoDuration 说明

功能：

获得视频总时长。

返回值：

视频总时长，类型 long，单位毫秒。

4.17 GetCurrentPosition 说明

功能：

获得时间当前进度。

返回值：

视频当前时间进度，类型 long，单位毫秒。

4.18 GetVideoWidth & GetVideoHeight 说明

功能：

获得视频宽度，高度。

返回值：

类型 long。

4.19 GetPlayerState 说明

功能：

获得当前播放状态。

使用：

为避免控制播放器出错，需要在播放、暂停、停止时判断播放状态。具体类型见“4.25.1 VideoPlayerState 播放状态”小节。

4.20 CreatVideoPlayer 说明

功能：

创建播放器，绑定事件监听与回调，获得 OPENGL 线程，创建视频矩阵。

使用：

仅调用一次创建方法，且必须在 PreparedPlayVideo 配置播放器所需的数据与初始化播放器之前调用。

4.21 OnVideoEvent 说明

底层监听到当前播放状态改变的回调消息函数，根据不同的回调事件响应不同的方法，具体回调事件见“4.24 delegate 监听与回调说明”。

4.22 OnExceptionEvent 说明

底层监听到视频准备时出现异常的回调消息获取函数，具体回调事件见“4.24 delegate 监听与回调说明”。

4.23 OnVolumeChangedEvent 说明

底层监听到设备音量改变时的回调消息获取函数，具体回调事件见“4.24 delegate 监听与回调说明”。

4.24 Delegate 监听与回调说明

1. OnReady：视频准备完成。视频准备完成后需要将视频纹理与播放器的 Material 绑定。此时可以开始播放视频；可以控制视频循环播放；可以设置视频总时长；

2. OnEnd：视频播放完毕。

3. OnVolumeChange：设备音量改变。

4. OnBufferProgressChange：更新已缓冲进度。参数：已缓冲进度占视频总时长的百分比。

5. OnBufferStart：开始缓冲。当前播放进度达到缓冲进度之后，将开始缓冲。

6. OnBufferFinish：缓冲结束。

7. OnProgressChange：当前播放进度。底层每 500 毫秒返回一次当前播放进度，参数为毫秒。

8. OnVideoError：视频准备异常。

9. OnSubtitleValueChange：字幕更新。需要调用 SetSubtitleSource 添加外部字幕，字幕文件需要与视频文件目录同级。

4.25 枚举类型说明

4.25.1 VideoPlayerState 播放状态

Idle：空闲状态，第一次创建播放器没有设置视频源

Preparing：正在准备视频播放源

Buffering：正在缓冲

Ready：视频源准备完成

Play：播放

Pause：暂停

Ended：播放完成

4.25.2 RenderCommand Render 显示指令

InitializePlayer：创建纹理

UpdateVideo：视频播放时，更新纹理

FreeTexture：释放纹理

4.25.3 VideoEvent 视频消息回调

VIDEO_READY = 0：SetDataSource 之后，视频准备完毕，可以开始播放了

VIDEO_ENDED = 1：视频播放完毕

VIDEO_BUFFER_PROGRESS = 2：视频缓冲进度

VIDEO_BUFFER_START = 3：视频卡顿时开始缓冲

VIDEO_BUFFER_FINISH = 4：缓冲完毕，可以继续播放了

VIDEO_PLAYING_PROGRESS = 5：播放进度

VIDEO_TEXTURE_CREATED = 6：纹理创建成功了(保证事件顺序：先 VIDEO_TEXTURE_CREATED 后 VIDEO_READY)

VIDEO_UPDATE_SUBTITLE = 11：更新字幕

4.25.4 ExceptionEvent

PATH_ERROR = 0：视频路径错误

NOT_SUPPORT_FORMAT = 1：不支持该视频格式

NOT_SUPPORT_SIZE = 2：不支持该视频尺寸(视频尺寸、码率或帧率太大了)

OTHER = 3：其他错误

EXCEPTION_NOT_BEST_SIZE = 4：视频不是最佳大小(最大支持视频尺寸：8192*4096，该尺寸下最佳码率 120M)

EXCEPTION_NETWORK_ERROR = 5：网络连接错误(只针对在线视频播放)

EXCEPTION_NOT_GOOD_SIZE = 6：视频超过最佳尺寸的 1.4 倍大小

其中，出现 PATH_ERROR、NOT_SUPPORT_FORMAT、NOT_SUPPORT_SIZE、OTHER、EXCEPTION_NETWORK_ERROR 这几种异常情况时，无法播放视频；出现 EXCEPTION_NOT_BEST_SIZE、EXCEPTION_NOT_GOOD_SIZE 这两种情况时，可以播放视频但是视频可能会丢帧导致播放卡顿。

5 GoogleVR API 接口函数

该 SDK 以 GVR SDK v1.40.0 为基础拓展，下面介绍部分接口，其余接口请前往官方网站(https://developers.google.com/vr/unity/reference/)获取详情。

5.1 GvrPointerInputModule 说明

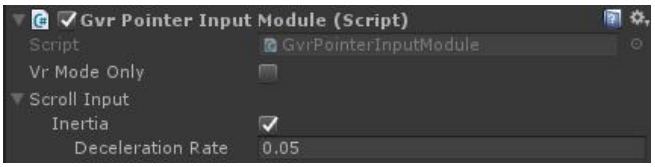

图 3.1 GvrPointerInputModule 设置示意图

功能：

GvrPointerInputModule 继承自 BaseInputModule，使用此脚本可以让基于 Canvas 的(UGUI)UI 元素和 3D 场景对象在应用程序中进行交互。所以 UI 元素在被 Pointer 选中或 Trigger ， Touching 触发的事件，可以传递出来。

使用：

进入 Project 选项卡，依次展开 Assets->SkyworthVR->GoogleVR->Prefabs->UI，将 GvrEventSystem 预制体放入场景替换原有的 EventSystem。

5.2 GvrPointerPhysicsRaycaster 说明

图 3.2 GvrPointerPhysicsRaycaster 设置示意图

功能：

GvrPointerPhysicsRaycaster 继承自 GvrBasePointerRaycaster，提供了一个用于 GvrPointerInputModule 的碰撞检测。

使用：

在场景中新建一个空对象，为其命名为 Player。将场景原有 Main Camera 拖动至 Player 之下。为 Main Camera 添加 GvrPointerPhysicsRaycaster 脚本。

5.3 StereoController 说明

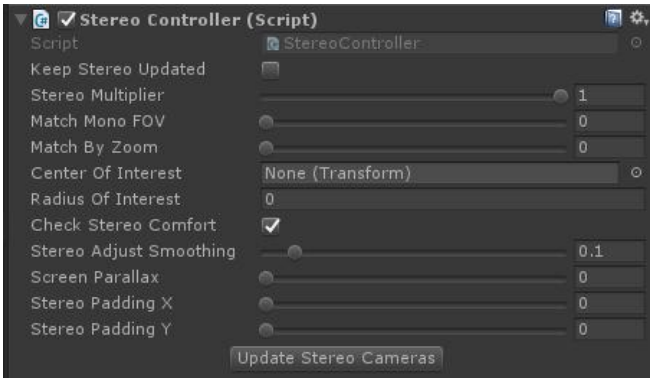

图 3.3 StereoController 设置示意图

功能：

绑定在 MainCamera 上，两个 GvrEye 渲染 stereo view 内容的控制器，这个脚本需要绑定到做 VR 渲染的摄像机上。

使用：

为 Main Camera 添加 StereoController 脚本。

5.4 GvrHead 说明

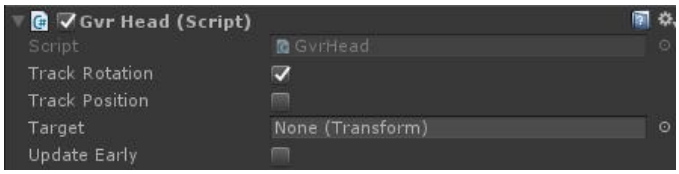

图 3.4 GvrHead 设置示意图

功能：

用户头部跟踪模拟，提供头部跟踪的数据给 Camera，在头部运动过程中它附着的 Transform 属性也会同步发生改变。

使用：

为 Main Camera 添加 GvrHead 脚本。

5.5 GvrHeadset 说明

功能：

GvrHeadset 是 VR 一体机耳机 API 的主要接口。一个场景中有且只能有一个 GvrHeadset 预制体。

使用：

进入 Project 选项卡，依次展开 Assets->SkyworthVR->GoogleVR->Prefabs->Headset，将 GvrHeadset 预制体放入场景即可。

5.6 GvrViewer 说明

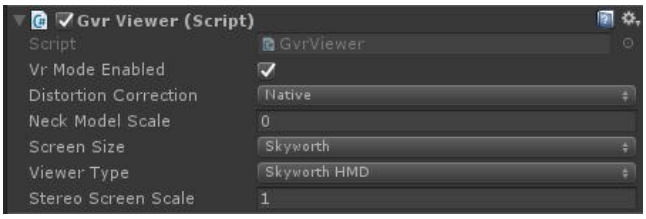

图 3.5 GvrViewer 设置示意图

功能：

用于初始化 Devices,在编辑器运行的时候会初始化 Editor 的 Devices。一个场景中有且只能有一个 GvrViewerMain 预制体。

使用：

进入 Project 选项卡，依次展开 Assets->SkyworthVR->GoogleVR->Prefabs，将 GvrViewerMain 预制体放入场景即可。

6 3DoF 手柄说明

SDK 提供 3DoF 手柄控制器的按键与触摸板响应事件，调用手柄相关接口请参照第三章 API 接口函数——GvrControllerInput 说明，这里介绍接口参数与物理手柄按键的对应关系。

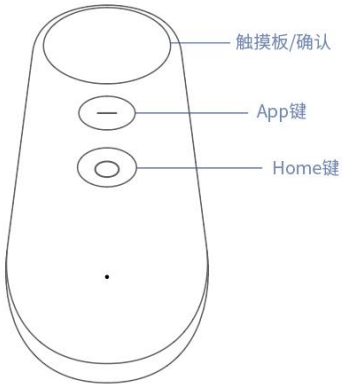

图 5.1 3dof 手柄示意图

手柄物理按键与 API 接口函数中参数的对应关系如下：

物理按键	键值	API 相关接口
触摸板/确认	1 << 0	ClickButton ClickButtonDown ClickButtonUp
App 键	1 << 1	AppButton AppButtonDown AppButtonUp
Home 键	1 << 3	HomeButtonDown HomeButtonState

7 凝视点击

SDK 中提供了倒计时出发点击的功能，当不需要手柄操作时可以使用凝视倒计时来触发点击事件。该功能需要和 GvrReticlePointer 一起使用。（注：当期 3DoF 手柄已成为 VR 一体机标准交互方式，如特定的原因，请默认使用 3DoF 手柄作为交互方式。）

7.1 使用说明

在 Assets->SkyworthVR->GoogleVR->Prefabs->UI 中将 SvrReticleDownClick 预制体放在 GvrReticlePointer 下面，然后在场景中创建一个 Button，点击运行。当锚点 Hover 到 Button 上时会出现倒计时效果。

SvrReticleDownClick 只会检查实现了 PointerClick 的物体才起作用，如果你不是使用的 PointerClick 将没有任何效果。

默认情况下倒计时是 1 秒，我们也提供了 UICountDown 组件来控制倒计时时间。

7.2 组件 UICountDown

当需要控制按钮倒计时的时间时可以将 UICountDown.cs 脚本挂在相应 PointerClick 的物体上，然后设置 Count 的值，单位是秒。