IDEAL OS SDK（Unity）

开发文档

IDEAL OS SDK（Unity）.....1

开发文档.....1

1 简介.....4

1.1 SDK 支持平台.....4

1.2 开发环境.....4

1.3 SDK 构成.....4

2 SDK 的使用说明.....4

2.1 SDK 导入步骤.....4

2.1.1 全新使用.....4

2.1.2 V1.1.x 或更早版本升级.....5

2.1.3 UGUI 版本升级.....5

2.2 IVRCamera 说明.....5

2.3 IVR Input Module 说明.....6

2.4 IVR Physics Raycaster 说明.....7

2.5 IVR Raycaster 说明.....8

2.6 项目设置.....8

2.7 导出到设备运行.....11

2.8 如何兼容 Gear VR.....11

3 用户操作输入.....12

3.1 IVR.IVRK2Event.....12

3.2 IVR.IVRH2Event.....12

3.3 IVRTouchPad.....13

4 V1.2.0 之前接口说明.....14

4.1 IVRCamera.cs 和 IVRPanel.cs -- 已弃用.....14

4.2 IVRTouchPad.cs -- 改为头控和 H2 云鼠通用事件接口.....14

4.3 IVRAnchorHandler.cs -- 已弃用.....14

4.4 AnchorWidget.cs 和 VREventListener.cs -- 暂时保留.....15

5 EventSystem 说明.....15

5.1 IVRInput.cs.....15

5.2 IVRRayPointerEventData.cs.....15

5.3 IOnHoverHandler 接口.....16

6 使用 IDEALENS H2 云鼠.....16

6.1 IVRHandlerInputModule 说明.....17

6-2 IVRHandlerEventData 说明.....18

6-3 IVRInputHandler 说明.....18

6-4 更多请参考 Handler_demo.....19

7 云端通信的使用及接口说明.....19
8 开发注意事项.....20

1 简介

1.1 SDK 支持平台

硬件设备：IDEALENS K2（即对应硬件平台为 7420，双 Camera 版本）。

1.2 开发环境

Unity：目前支持 Unity4.6.3f1 到 Unity5.6.1，推荐使用 Unity5.3.4f1 版本。（如果其他版本有兼容性问题，请联系我们）
（Unity5.3.5f1 有兼容性问题，会导致 UI 无法显示，请不要使用）
（Unity4.6.9f1 和 5.6.2 以上的版本有 bug，上真机无法运行，请不要使用）
Android： 6.0 版本。
Java： 1.7 及以上版本。

1.3 SDK 构成

获取 SDK 安装压缩包 ivrsdk.rar，解压后可看到 ivrsdk.unitypackage 文件。

2 SDK 的使用说明

2.1 SDK 导入步骤

解压 SDK 安装压缩包 ivrsdk.rar，根据以下 3 种情况，按步骤导入 ivrsdk.unitypackage。

2.1.1 全新使用

1. 将 ivrsdk.unitypackage 导入 Unity 工程中。
2. 在 Unity 菜单栏选择 IVRSDK -> Create IVRCamera Component.
3. SDK 使用 Unity 的 EventSystem 控制所有的事件，选择 IVRSDK -> Create IVREventSystem 创建使用 IVRInputModule 的 EventSystem.
4. 如果需要使用物理射线作为 EventSystem 的触发射线，选择 IVRSDK -> Use IVR Physics Raycast.
5. 如果需要使用 UGUI 的事件触发，选中场景中的 Canvas 物体，点击 IVRSDK -> Use IVR Graphic Raycast.

（4，5 步不是必须的，开发者也可自己定义射线作为 EventSystem 的输入，具体方法详见后文）

2.1.2 V1.1.x 或更早版本升级

使用老版本的 sdk 的升级步骤：

1. 备份好将要升级的 Unity 工程。
2. 打开一个新场景。
3. 删除工程中原有的 IVRSDK 目录，在 Plugins 目录中删除 libidealseePlugin.so 和 vrlib.jar/vrlib-idealsee.jar。（确保该目录中不包含第三方的代码和资源）
4. 导入新 SDK 中的 ivrsdk.unitypackage。
5. 打开工作场景。
6. 点击 IVRSDK -> Create IVRCamera Component。
7. 点击 IVRSDK -> Create IVREventSystem。
8. 点击 IVRSDK -> Use IVR Physics Raycast。
9. 选择新创建出来的 EventSystem，修改 IVRInputModule 的配置，勾选 Use IVR Hover 和 Send Event To All 选项。
10. 删除弃用的 IVRTouchPad，IVRResetPos，IVRPanel 和 IVRCamera。
11. 检查场景中是否有绑定关系丢失，运行检查是否有报错。

2.1.3 UGUI 版本升级

之前只用 IVRSDK_UGUI 版本的升级步骤

1. 备份好将要升级的 Unity 工程。
2. 打开一个新场景。
3. 删除工程中原有的 IVRSDK 目录。（确保该目录中不包含第三方的代码和资源）
4. 导入新 SDK 中的 ivrsdk.unitypackage。
5. 打开工作场景。
6. 移除 IVRCamera 上丢失关联的组件，点击 IVRSDK -> Create IVRCamera Component，重新配置 IVRManager 选项。
7. 检查场景中是否有绑定关系丢失，运行检查是否有报错。

2.2 IVRCamera 说明

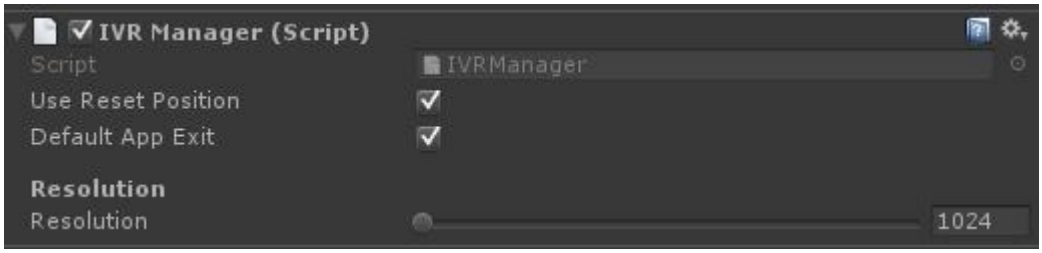

图 2-1 IVRManager 配置

IVRCamera 对象是 SDK 中的重要组件，RightEye 和 LeftEye 是两个 Camera，用来选择左右两眼看到的图像，Anchor 作为两眼的中点可以用来放置锚点和作

为射线发射原点（为了方便发射射线，Anchor 上有一个未激活的 Camera，这是默认配置，请不要修改！）。IVRCamera 上有 IVRManager 组件，上面有两个参数：

Use Reset Position：是否使用重置视角功能，如果勾选，当用户长按触摸板的时候就会触发重置视角功能。

Default App Exit：是否使用默认的返回键退出应用，如果勾选，当用户点击设备上的返回键时会退出程序。如果开发者想自己处理返回键点击事件，一定不能勾选这个选项。

Resolution: 左右眼输入的 RenderTexture 的分辨率，数值越大，画面越清晰，但系统开销越大，开发者可以根据实际情况进行调节。

注意：

1. 场景中不能有其他 Camera，其他 Camera 的绘制也不会显示在设备上，请将其删除。

2. 在 Unity 编辑器和设备上 IVRCamera 工作是不一样的，在编辑器中鼠标模拟的头部转动直接作用于 IVRCamera 对象，而在设备上传感器的旋转作用于 LeftEye 和 RightEye，而 IVRCamera 不会旋转。同时，在编辑器上，LeftEye 和 RightEye 位置重合，而在设备上会在应用程序初始化的时候根据默认的瞳距和头部高度初始化 LeftEye 和 RightEye 的位置。

2.3 IVR Input Module 说明

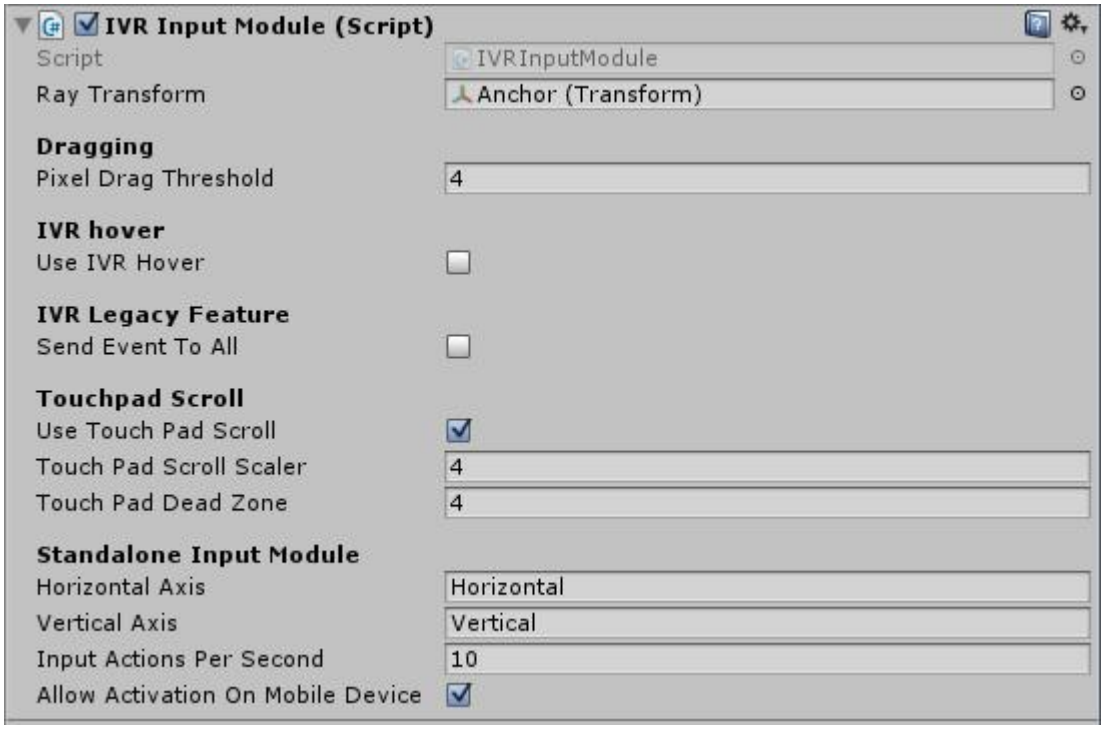

图 2-2 IVRInputModule 配置

从 v1.2.0 版本开始，SDK 都将默认使用 Unity 原生的 EventSystem 来处理所有的事件，IVRInputModule 是基于 Unity 的 PointerInputModule 扩展而来，作为 SDK 的默认输入模块，参数如下：

Ray Transform：在 VR 应用中默认使用射线触发事件，IVRInputModule 输入模块需要知道射线的发射原点。默认是 IVRCamera 下的 Anchor，也可以自己定义，该参数不能为空。
Pixel Drag Threshold：触摸板触发滑动的阈值。
Use IVR Hover：为了兼容原有系统中 VREventListener 中的 IsHover 参数，增加了 IOnHoverHandler 接口，不勾选则不会触发 IOnHoverHandler 接口。
Send Event To All：UGUI 的机制是默认只向最近的一个碰撞到的物体发送事件，勾选这个选项后将向所有碰撞到的物体发送事件。
Use Touch Pad Scroll：是否使用在 touchpad 上的滑动作为滚轮参数。不勾选则不会触发滚轮事件。
Touch Pad Scroll Scaler：touchpad 上的滑动位移到滚轮参数的转换系数。
Touchpad 的原始位移会除以这个系数后得出滚轮参数。
Touch Pad Dead Zone：touchpad 上的滑动位移小于该值时不触发滚轮事件。
Standalone Input Module：配置参数为继承的 PointerInputModule 参数，如无必要，请勿修改。尤其注意如果 Allow Activation On Mobile Device 没有勾选的话将在设备上无法工作！

2.4 IVR Physics Raycaster 说明

图 2-3 IVRPhysicsRaycaster 配置

IVRPhysicsRaycaster 继承自 BaseRaycaster，是用来检测和 Collider 的物理碰撞的，必须和 Camera 绑定在一个 GameObject 上（默认绑定在 Anchor）。Event Mask 可以用来选择碰撞检测哪些 Layer 层级。
注意：如果 Collider 上没有勾选 IsTrigger 选项，该碰撞体后的物体都不会触发碰撞事件。

2.5 IVR Raycaster 说明

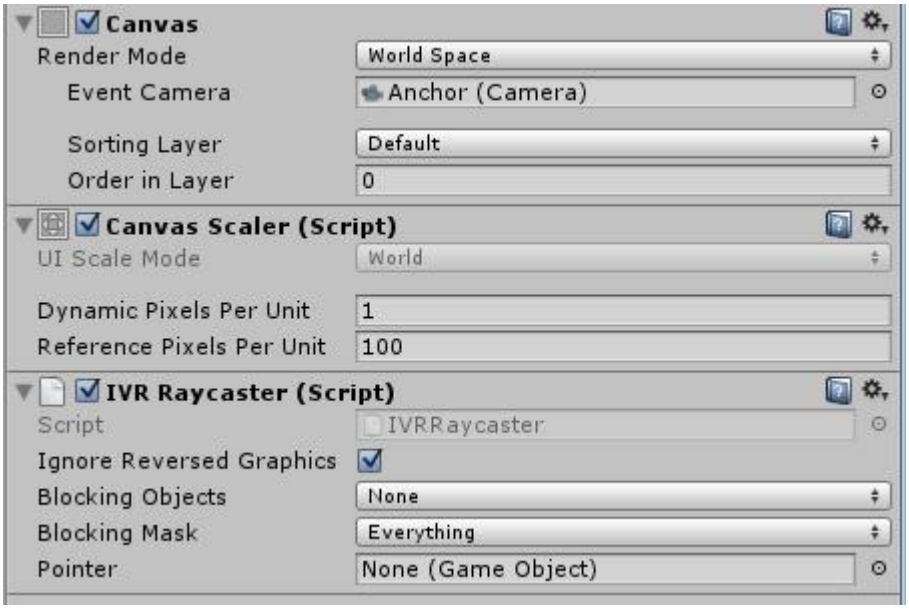

图 2-4 IVRRaycaster 配置

IVRRaycaster 继承自 GraphicRaycaster，是用来检测 UGUI 的碰撞的，必须和 Canvas 绑定在一个 GameObject 上且该 Canvas 必须设置一个 Event Camera（默认是 Anchor）。配置参数均和 GraphicRaycaster 一致，可以直接查阅官方文档。

2.6 项目设置

1. 导出设置为横屏，具体参数如下图 2-5 所示：

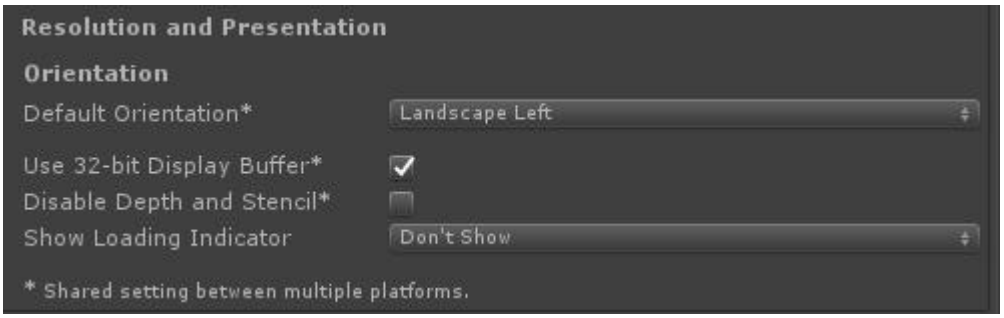

图 2-5 导出设置示意图

2. Build Settings 设置，Texture Compression 选择为 ETC2(GLES3.0)如图 2-6 所示:

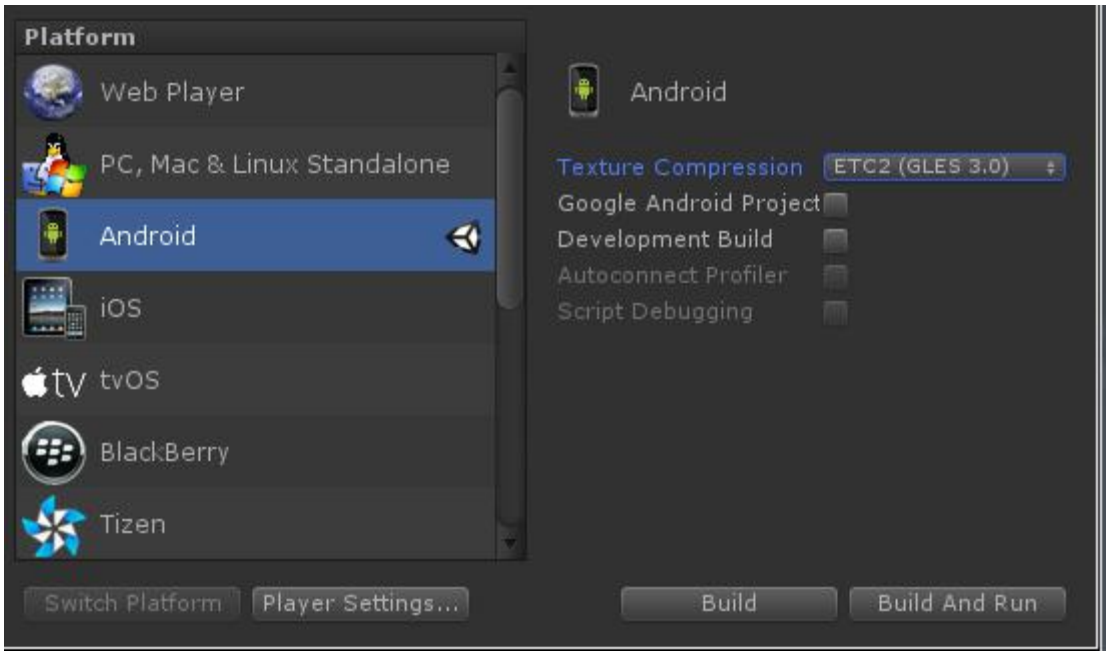

图 2-6　Build Settings 设置示意图

3. PlayerSettings 设置，不要勾选 Virtual Reality Supported。具体参数如下图 2-7 所示：

图 2-7　PlayerSettings 设置示意图

4. 在 PlayerSettings 中打开读写 SD 卡的权限（重要）。如下图

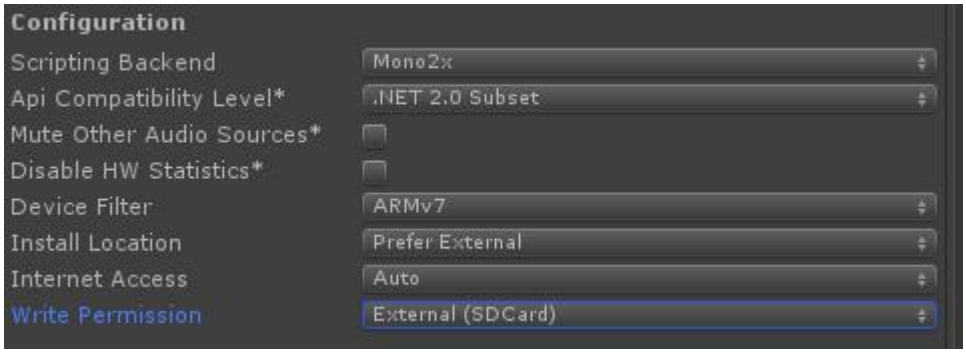

图 2-8　读写 SDCard 权限

5. QualitySettings 设置，具体参数如下图 2-9 所示：

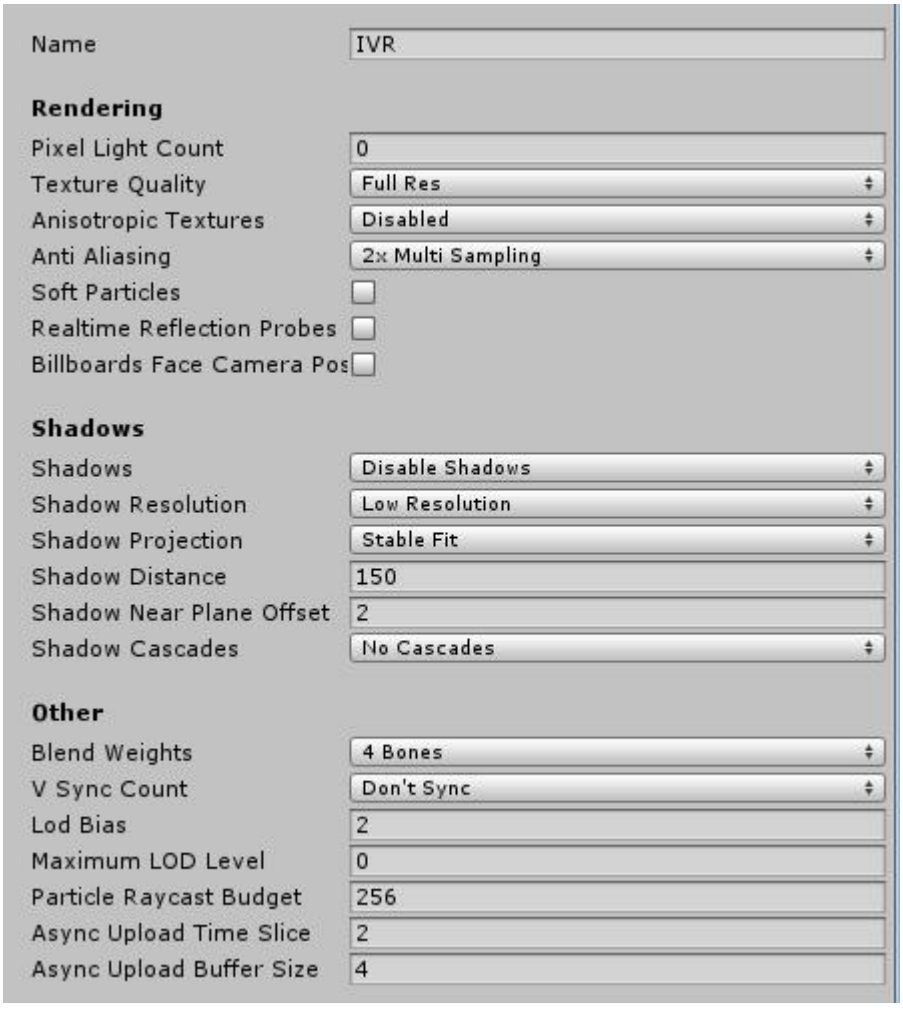

图 2-9　QualitySettings 设置示意图

Anti Aliasing 可以根据需要进行调整。

V Sync Count 一定不能开启，必须选择 Don’t Sync。

2.7 导出到设备运行

1. 将设备通过 USB 连接到电脑，当连上电脑后，Launcher 界面的电池图标会变成充电状态，有个闪电图标在电池上。
2. 点击 File -> Build & Run，等进度条结束就 OK 了！

2.8 如何兼容 Gear VR

开发者如果使用 Unity5.0~Unity5.2 版本开发 Gear VR 应用，请按照如下步骤即可在应用中使用 Ideal OS SDK，发布在 IDEALENS K2 平台上：

1. 导入 SDK。
2. 删除场景中的 OVRPlayerController 或者 OVRCameraRig。
3. 使用工具栏创建 IVRCamera 和 EventSystem，根据需要选择使用不同的 Raycaster。

4. Build&Run。

开发者如果使用 Unity5.3 开发 Gear VR 应用，请按照如下步骤即可在应用中使用 Ideal OS SDK，发布在 IDEALENS K2 平台上：

1. 导入 SDK。
2. 删除场景中的 Main Camera。
3. 使用工具栏创建 IVRCamera 和 EventSystem，根据需要选择使用不同的 Raycaster。
4. 在 PlayerSettings 设置中开发者请不要勾选 Virtual Reality Supported，如下图 2-10 所示：

图 2-10　PlayerSettings 中 Virtual Reality Supported 设置示意图

5. Build&Run。

3 用户操作输入

为了最大限度的兼容 IDEALENS H2 云鼠的输入并且提供新接口区分原有的头控触摸板输入，从 v1.3.2f1 版本开始，SDK 中提供了三个模块给开发者注册相关的输入事件的回调。

3.1 IVR.IVRK2Event

头控输入事件接口，包含以下事件：

`event` TouchEvent_HomeButton
头控返回键点击事件。

`event` TouchEvent_onLongPress
头控触摸板长按事件，长按触摸板超过 1 秒并且没有发生较大滑动位移时触发。

`event` TouchEvent_onPress
头控触摸板持续触摸事件，触摸发生时每一帧都会发送该事件。

`event` TouchEvent_onPressDown
头控触摸按触摸事件，触摸发生那一帧发送该事件。

`event` TouchEvent_OnPressUp
头控触摸板触摸离开事件，触摸停止那一帧发送该事件。

`event` TouchEvent_onScroll
头控触摸板滑动事件，当手指在触摸板滑动时，发送每一帧滑动的距离。

`event` TouchEvent_onSingleTap
头控触摸板点击事件，当手指触摸触控板并在 1 秒内离开触摸板，则会触发点击事件。

`event` TouchEvent_onSwipe
头控触摸板翻页事件，当手指在触摸板上滑动超过了触摸板的 20%时触发翻页事件，参数为翻页方向 SwipEnum.MOVE_FOWRAD(向前），SwipEnum.MOVE_BACK(向后），SwipEnum.MOVE_UP(向上），SwipEnum.MOVE_DOWN(向下）。

该模块还有一个接口：`void` Update(float fCurrentUnscaledTime)，需要被每帧调用并传入一个表示系统时间的参数，如果开发者使用了 IVRInputModule 或 IVRHandlerInputModule 作为 EventSystem 的输入模块，则不需要再调用该接口。

3.2 IVR.IVRH2Event

H2 云鼠输入事件系统，包含以下事件：

`event` ButtonEvent_onClick
云鼠按键点击事件，参数为按键标志位（具体参照 6.3 中 ControllerButton 定义）。当云鼠按键被按下并在 2 秒内释放，则触发一次云鼠按键点击事件。

`event` ButtonEvent_onLongPress
云鼠按键长按事件，参数为按键标志位，当云鼠按键被按下超过 2 秒则触发长按事件。

`event` ButtonEvent_onPress

云鼠按键持续按下事件，参数为按键标志位，当云鼠按键被按下后，直至释放前，每一帧都会发送该事件。

event ButtonEvent_onPressDown

云鼠按键按下事件，参数为按键标志位，云鼠按键被按下那一帧发送该事件。

event ButtonEvent_onPressUp

云鼠按键抬起事件，参数为按键标志位，云鼠按键抬起释放那一帧发送该事件。

event ButtonEvent_onScroll

云鼠触摸板滑动事件，当手指在云鼠触摸板上滑动时，每一帧发送该事件，参数为滑动距离偏移量。

event ButtonEvent_onSwipe

云鼠触摸板翻页事件，当手指在云鼠触摸板上滑动超过 20%时触发翻页事件，参数为翻页方向（跟头控一致）。

同 IVRK2Event，该模块也有一个接口 void Update(float fCurrentUnscaledTime)，如果开发者使用了 IVRInputModule 或 IVRHandlerInputModule 作为 EventSystem 的输入模块，则不需要再调用该接口，如果开发者没有使用该接口，开发者需要在每一帧更新了云鼠数据后（具体参照 6.3 章节），调用该接口。

3.3 IVRTouchPad

开发者可以通过该接口同时兼容头控和 H2 云鼠输入事件，包含以下事件：

event TouchEvent_onLongPress

长按事件，条件为触发了头控触摸板长按事件或 H2 云鼠的 APP 键长按事件。

event TouchEvent_onPress

持续按下事件，条件为触发了头控触摸板持续按下事件或 H2 云鼠的触摸板按键或扳机键的长按事件。

event TouchEvent_onPressDown

按下事件，条件为触发了头控触摸板按下事件或 H2 云鼠的触摸板按键或扳机键的按下事件。

event TouchEvent_OnPressUp

抬起事件，条件为触发了头控触摸板的抬起事件或 H2 云鼠的触摸板按键或扳机键的抬起事件。

event TouchEvent_onScroll

滑动事件，条件为触发了头控触摸板的滑动事件或 H2 云鼠的触摸板的滑动事件。

注意：由于硬件差异原因，头控触摸板和 H2 云鼠触摸板的 x，y 轴并不一致，请开发者注意。

event TouchEvent_onSingleTap

点击事件，条件为触发了头控触摸板的点击事件或 H2 云鼠的触摸板按键或扳机键的点击事件。

event TouchEvent_onSwipe

翻页事件，条件为触发了头控触摸板的翻页事件或 H2 云鼠的翻页事件。

另外 IVRTouchPad 还有 5 个接口方法：

`void` void Active()
只有调用了该接口后，IVRTouchPad 的事件才会生效，开发者如果使用了 IVRInputModule 或 IVRHandlerInputModule 作为 EventSystem 的输入模块，则不需要再调用该接口。
`void` Deactive()
调用该接口后，IVRTouchPad 的事件将不再生效。
`void` AddKeyEvent(BoolDelegate _delegate, KeyLayout layout)
注册一个回调函数_delegate，当头控返回键按下或 H2 云鼠的 HOME 键按下后按照添加顺序和层级逐个调用注册的回调函数。Layout 共 7 个层级，层级越高的回调会优先调用，同一层级按照注册先后顺序调用。一个层级中如果有回调返回 true，则不会调用下一层级的回调。
`void` RemoveKeyEvent(BoolDelegate _delegate, KeyLayout layout)
取消注册回调函数_delegate。
`void` ClearEvent()
清空所有通过 AddKeyEvent 注册的回调函数。

4 V1.2.0 之前接口说明

V1.2.0 之前的 SDK 没有使用 Unity 的 EventSystem，而是在 IVRAnchorHandler.cs 中触发 IVRCamera.cs 发射射线去进行物理碰撞检测，AnchorWidget.cs 作为所有 UI 事件接收方的基类去处理所有 IVRAnchorHandler.cs 和 IVRPanel.cs 发送的事件，VREventListener.cs 继承了 AnchorWidge.cst 方便用户调用。而触摸板和返回按钮的监听实现在 Android 层，然后发送消息通知到 IVRTouchPad.cs。
V1.2.0 开始 SDK 全面转向使用 Unity 的 EventSystem，并且不再在 android 层监听触摸板和返回键事件，而是直接调用 Unity 的原生接口，所以之前的接口部分被废弃，部分考虑到兼容性的问题暂时保留。
对于新使用 SDK 的项目，以下接口都不建议使用，请直接使用 UGUI。

4.1 IVRCamera.cs 和 IVRPanel.cs -- 已弃用

IVRPhysicsRaycaster.cs 代替了该模块功能。使用之旧版本 SDK 的在更新后需要将其从场景中删除，IVRCamera.cs 原来默认绑定在 Anchor 上。

4.2 IVRTouchPad.cs -- 改为头控和 H2 云鼠通用事件接口

详见章节 3.3。

4.3 IVRAnchorHandler.cs -- 已弃用

EventSystem 代替了该模块功能。使用之旧版本 SDK 的在更新后需要将其从场景中删除。

4.4 AnchorWidget.cs 和 VREventListener.cs -- 暂时保留

所有接口仍然可用，通过实现大部分原生的 IEventSystemHandler 接口和新增的 IOnHoverHandler 接口来兼容 EventSystem。使用旧版 SDK 的不需要做任何其他改动。

5 EventSystem 说明

EventSystem 是 Unity 用来处理用户输入交互的核心模块，它需要一个 InputModule 来配合工作处理用户的输入，开发者可以通过继承 BaseInputModule 类来实现自己的输入模块。Ideal OS SDK 从 v1.2.0 开始全面兼容 Unity 的 EventSystem，通过 IVRInputModule，IVRPhysicsRaycaster 和 IVRRaycaster（这三个模块在 2.3，2.4，2.5 章节有介绍）来实现用户在 IDEALENS K2 设备上的操作和交互。
除了上述三个模块，还有另外两个重要的模块和一个新增的事件接口下面做一下介绍。

5.1 IVRInput.cs

设备上的触摸板和返回按键可以直接使用 Unity 的原生接口读取。
触摸板事件：

```
if (AndroidInput.touchCountSecondary > 0)
{
    Touch t = AndroidInput.GetSecondaryTouch(0);
}
```

AndroidInput 可以获取的 Touch 事件和 Input 接口一致。
返回键事件可以用 `Input.GetKeyDown(KeyCode.Escape);`获取。
为了方便调用和在编辑器模拟事件，所以 IVRInput.cs 将以上两个接口进行了封装，开发者可以选择使用这个接口或者直接调用原生接口。

`IVRInput.GetKeyDown()`

触摸板按下事件

`IVRInput.GetKeyUp()`

触摸板抬起事件

`IVRInput.IsHomeButtonDown()`

返回键按下事件

`IVRInput.mousePosition`

当前触摸板触发位置

在 Unity 编辑器内，可以用空格键和鼠标左键模拟触摸板点击事件。ESC 键可以模拟返回键。上下左右方向键可以模拟触摸板滑动事件。

5.2 IVRRayPointerEventData.cs

继承自 PointerEventData，保持原有接口不变，新增了一些变量：

```
public List<Vector3> HitPoints;
public List<GameObject> HitResults;
```

```
public List<GameObject> PressObjects;
public Vector2 TouchPadPosition;
public Ray worldSpaceRay;
```

HitPoints：射线击中物体命中点的世界坐标。按照由近及远的顺序排列。
HitResults：射线击中物体，按照由近及远的顺序排列。
PressObjects：触摸板按下时所有接收到 PressDown 事件的物体，按照由近及远的顺序排列。
TouchPadPosition：PointerEventData 原生记录的 position 和 pressposition 都是相对于屏幕的坐标，在 VR 中即是屏幕中点坐标，所以需要该变量记录下当前触摸板被触发的坐标。
worldSpaceRay：当前用于检测碰撞的射线。

5.3 IOnHoverHandler 接口

接口方法为：`void OnHover(BaseEventData eventData);`
因为 UGUI 只有 IPointerEnterHandler 和 IPointerExitHandler 在射线进入和离开物体才触发的两个事件，无法实时更新物体的碰撞点和实现其他一些功能，所以添加了此接口，该事件会在射线碰撞到物体后每一帧都发送该事件，离开物体后也会再发送一次该事件。例（AnchorWidget 中的实现）：

```
void IOnHoverHandler.OnHover(BaseEventData eventData)
{
    IVRRayPointerEventData pointer = eventData as IVRRayPointerEventData;
    if (pointer == null) return;
    if (pointer.HitResults.Contains(gameObject))
    {
        hitPoint = pointer.HitPoints[pointer.HitResults.IndexOf(gameObject)];
        if (!isHover) RunOnHover(true);
    }
    else if(isHover)
    {
        RunOnHover(false);
        LoseFocus();
    }
}
```

注意：必须在 IVRInputModule 中勾选 Use IVR Hover 才会发送该事件！（详见 2.3 章节）

6 使用 IDEALENS H2 云鼠

在新的 IDEALENS K2 设备上增加了对 IDEALENS H2 云鼠的支持（20161225 之后的 ROM），开发者可以使用 v1.3.2f1 版本及以后的 SDK 方便的获取到云鼠操作。只需要简单替换之前的 IVRInputModule 为新增的

IVRHandlerInputModule 作为 EventSystem 的输入即可接入 H2 云鼠操作，再进行简单设置就能将 H2 云鼠操作映射为原有的 K2 的触摸板操作。
注意：由于硬件改版，v1.3.1 中扳机按键为霍尔器件，所以有专门接口返回按键键值，在 v1.3.2f1 中，扳机板件视为同其他按键一样的普通按键，所以在 `ControllerButton` 中增加 `CONTROLLER_BUTTON_TRIGGER` 位。同时为了保证兼容性，原有的 trigger 相关接口都会保留且仍有效（键值只有 0 和 255），但是建议开发者还是尽早升级使用新接口。

6.1 IVRHandlerInputModule 说明

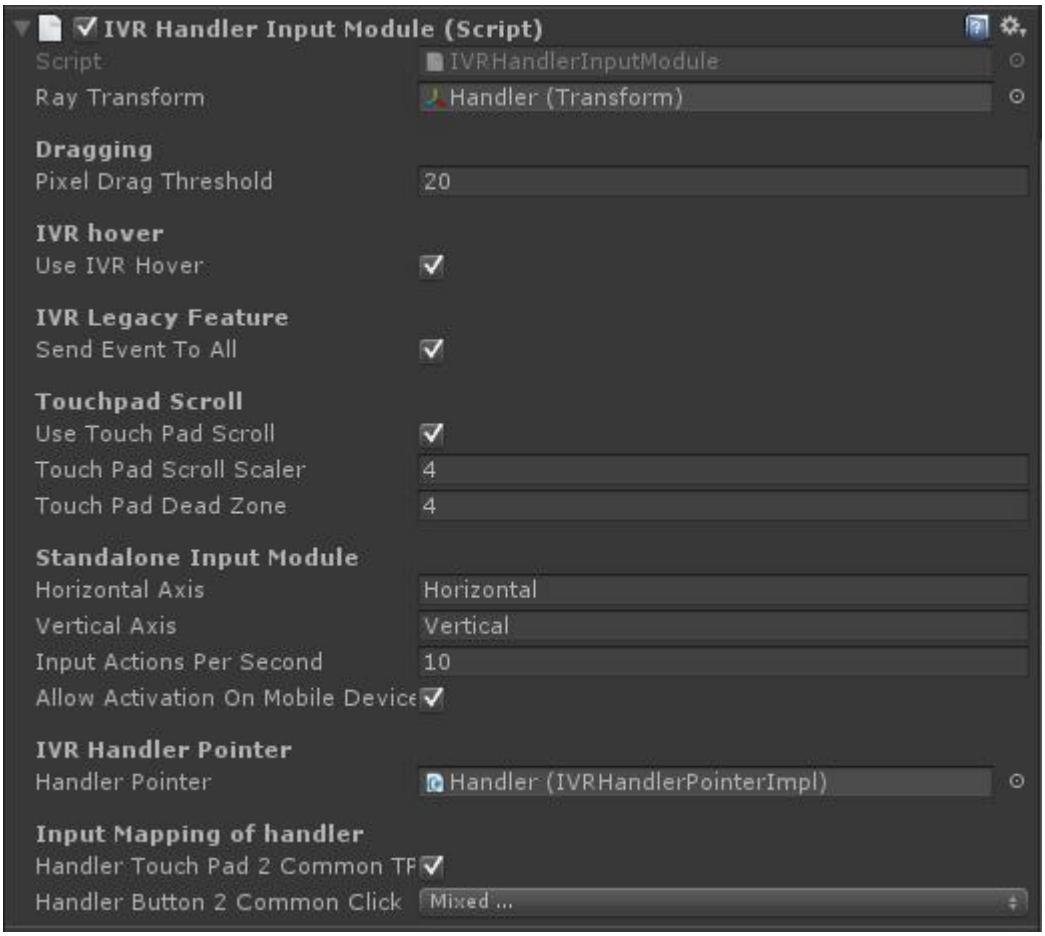

图 6-1 Handler_demo 场景中的 IVRHandlerInputModule

IVRHandlerInputModule 继承自 IVRInputModule，添加了对 H2 云鼠的支持，如果需要使用 H2 云鼠，请用 IVRHandlerInputModule 替换原来的 IVRInputModule。如图 5-1 所示，IVRHandlerInputModule 上半部分跟 IVRInputModule 一致，新增了 IVR Handler Pointer 和 Input Mapping of Handler 两个设置。IVR Handler Pointer 可以指定跟随 H2 云鼠姿态变化的射线（具体可以参考 Handler_demo 样例场景），也可以为空。Input Mapping of Handler 用来设置 H2 云鼠操作是否映射原来 K2 设备上的触摸板操作。
Handler Touch Pad 2 Common TP -- 勾选则将 H2 云鼠上的触摸板操作映射为 K2 上的触摸板操作发送给 UI 层。

Handler Button 2 Common Click -- 选择哪些 H2 云鼠按键映射为 K2 上的轻触点击操作发送给 UI 层。
请开发者注意区分 UI 事件系统和第三章节介绍的用户输入事件系统的异同。

6-2 IVRHandlerEventData 说明

继承自 IVRRayPointerEventData，如果使用了 IVRHandlerInputModule，那么系统中传递的事件都为 IVRhandlerEventData。比 IVRRayPointerEventData 增加了 H2 云鼠的相关信息：

```
short HandlerButtons;
```

H2 云鼠按键位

```
Vector3 HandlerPose;
```

H2 云鼠姿态（欧拉角）

```
Vector2 HandlerTouchPosition;
```

H2 云鼠触摸板触摸坐标（0，0）表示没有触摸

6-3 IVRInputHandler 说明

H2 云鼠控制类，方便开发者直接控制 H2 云鼠和读取相关数据，接口如下：

```
static void Activate();
```

激活 H2 云鼠操作，会自动被 IVRHandlerInputModule 调用，如果开发者不使用 IVRHandlerInputModule 作为 EventSystem 的输入模块，需要手动调用该方法激活云鼠。

```
static void Deactivate();
```

关闭 H2 云鼠操作，会自动被 IVRHandlerInputModule 调用，如果开发者不使用 IVRHandlerInputModule 作为 EventSystem 的输入模块，当不再需要使用云鼠后，调用该方法。

```
static void UpdateModule();
```

更新 H2 云鼠操作信息，会每帧被 IVRHandlerInputModule 调用，如果开发者不使用 IVRHandlerInputModule 作为 EventSystem 的输入模块，需要手动每帧调用该方法，以获取最新的数据。

注意：如果 H2 云鼠连接状态不正确（IsConnected()接口返回 false），则调用该接口无效。

注意：该方法最好在使用 H2 云鼠数据之前调用，以保证数据的有效性。

```
static void OnPause(bool pause);
```

当程序暂停和恢复时需要调用该接口，会自动被 IVRHandlerInputModule 调用。如果开发者不使用 IVRHandlerInputModule 作为 EventSystem 的输入模块，建议在程序暂停和恢复时调用该接口，以保证程序恢复时云鼠数据的有效性和程序休眠时云鼠可以待机休眠。

```
static void CheckState();
```

更新 H2 云鼠最新连接状态。注意：H2 云鼠连接状态不会自动更新，开发者需要时时关注 H2 云鼠状态是否可用。由于该操作为耗时操作，不建议每一帧在主线程调用，可以使用协程定时检查。如果开发者使用 IVRHandlerInputModule 作为 EventSystem 的输入模块，则不需要考虑该问题。

```
static bool IsConnected();
```

获取 H2 云鼠的连接状态。

注意：只有调用 CheckState 方法可以更新该状态。

```
static bool GetKey(int iCode);
```

```
static bool GetKey(ControllerButton kCode);
static bool GetKeyDown(ControllerButton kCode);
static bool GetKeyDown(int iCode);
static bool GetKeyUp(int iCode);
static bool GetKeyUp(ControllerButton kCode);
```

以上是判断 H2 云鼠按键状态的接口，H2 用一个 short 字符表示按键信息，每个按键对应一位 bit，具体如下：

```
[Flags]
    public enum ControllerButton
    {
        CONTROLLER_BUTTON_MENU = 1,
        CONTROLLER_BUTTON_HOME = 2,
        CONTROLLER_BUTTON_VOL_UP = 4,
        CONTROLLER_BUTTON_VOL_DOWN = 8,
        CONTROLLER_BUTTON_TP_CLICK = 16,
        CONTROLLER_BUTTON_TRIGGER = 32,
    }
static Vector2 GetPosition();
static bool GetTouchpadDown();
static bool GetTouchpadUp();
```

获取 H2 云鼠触摸板的触摸信息，返回(0,0)表示没有触控触摸板。触摸开始一帧为 TouchPadDown，触摸结束一帧为 TouchPadUp。

```
static Vector3 GetRotation();
```

获取 H2 云鼠的姿态信息（欧拉角）。

```
static void SendVibrateCmd(int type);
```

向 H2 云鼠发送震动指令，0 为不震动，1 为震动。

6-4 更多请参考 Handler_demo

SDK 中新增了一系列样例场景，用来展现增加 H2 云鼠后，使用 H2 云鼠来控制激光射线替代原有的头部锚点交互设计。

7 云端通信的使用及接口说明

SDK 中包含和 IDEALENS 云端通信的接口，当用户在设置-账号界面使用自己的 IDEALENS 账号进行了登录以后，开发者可以在自己的应用内通过 SDK 的接口获取到用户的登录状态和账户信息。以下接口都包含在 IVR 这个命名空间内：

- ISDKUserInfo ISDK.Instance.GetAccountInfo()

获取账户信息，返回类 ISDKUserInfo，包含手机邮箱等所有用户信息

- bool ISDK.Instance.IsLogin()

判断是否登录,true 登录成功,false 未登录

8 开发注意事项

1. Ideal OS SDK 支持 Unity 原生蓝牙控制开发接口，开发者均可参考 http://docs.unity3d.com/Manual/index.html 文档进行开发。
2. Unity 默认设置启动的时候会显示 Unity 的 LOGO，但是在设备上没有进行分屏显示，体验非常不好，所以开发者需要将启动画面设置为纯黑色：

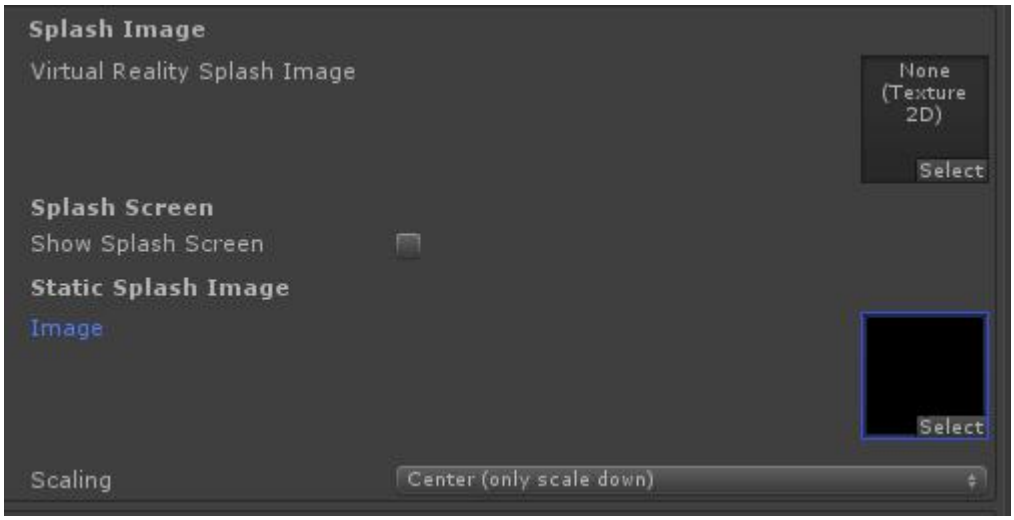

图 8-1 启动画面配置

3. 直接使用 Unity 默认的 Android 主题的应用，在启动的时候会闪一下白屏，用户体验非常不好。开发者可以手动修改一下 Unity 默认的配置文件：Unity\Editor\Data\PlaybackEngines\AndroidPlayer\Apk\res\values\styles.xml，将 android:Theme.Light.NoTitleBar.Fullscreen 修改为 android:Theme.Black.NoTitleBar.Fullscreen.
4. 开发者不要使用任何其他 Camera，这些 Camera 都不会在设备上显示出来。
5. 不能使用 Canvas 的 Screen Space - Overlay 模式绘制 UGUI，这样无法在设备上显示，请直接使用 World Space 模式。
6. 如果设备连接 USB 后无法识别，请确认安装或更新了最新的 Google USB Driver：http://developer.android.com/sdk/win-usb.html。如果仍然不能识别可以尝试安装 Samsung USB driver。有些 windows 版本仍然不能正确是识别硬件，如果在设备管理器中发现有黄色感叹号，可以尝试安装 Android Composite ADB Interface 解决该问题。

如有其它疑问或反馈请发送邮件至 mike@idealens.com。

感谢您对 Ideal OS SDK 的支持！