# ChangeLog

1. 增加了windos版和editor版的支持
2. 删除了NGUI依赖，使用UGUI制作Demo
3. 增加了平台依赖的代码，根据不同平台灵活处理权限问题
4. Android平台不再默认root权限，优先使用UsbManager打开设备
5. 重新整理了项目的结构

# Known Issue

1. Editor版本在某些机器上，停止的时候Unity会无反应

解决方案：避免使用Editor版本，直接build为windows版

1. Windows版本打印Log的时候会crash，而且导致摄像头无法关闭

解决方案：插拔摄像头电源，再次部署构建试试

1. 其他

# 1. 快速开始

**Step1 :** 将imiUnitySDK\_v0.1.2.unitypackage，任意导入到一个新的Unity工程中，导入全部资源

**Step2:** 打开Iminect/Demo/ImiExplore.unity的场景

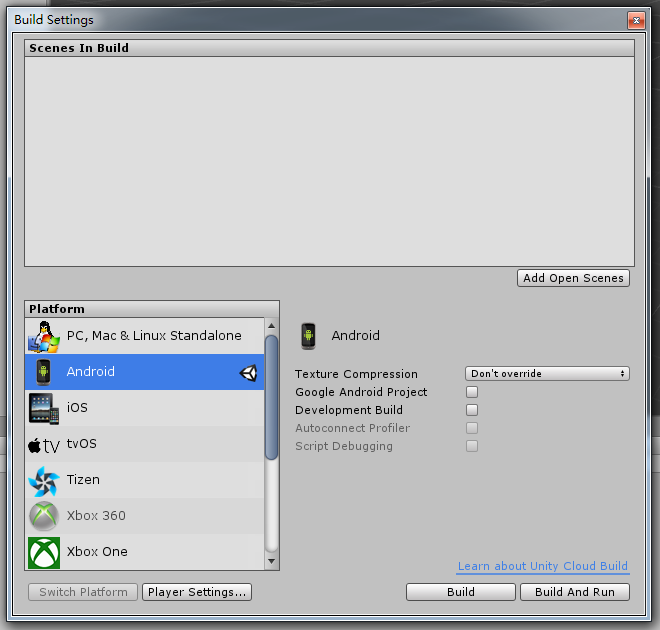
**Step3:** 将摄像头和设备连接好，启动设备

注：

1）如果是Android版，无需安装驱动，但要保证关联的设备可以通过电脑进行adb调试

2）如果是Windows版，则需要安装win下面的驱动，驱动文件在本目录下。安装好驱动后，打开设备管理器，检查是否安装成功（成功安装后，设备管理器列表中没有黄色的警告标志）

**Step4:** 点击File ----> Build Setting... , 在弹出框中选择PC/Android, 然后点击Build And Run

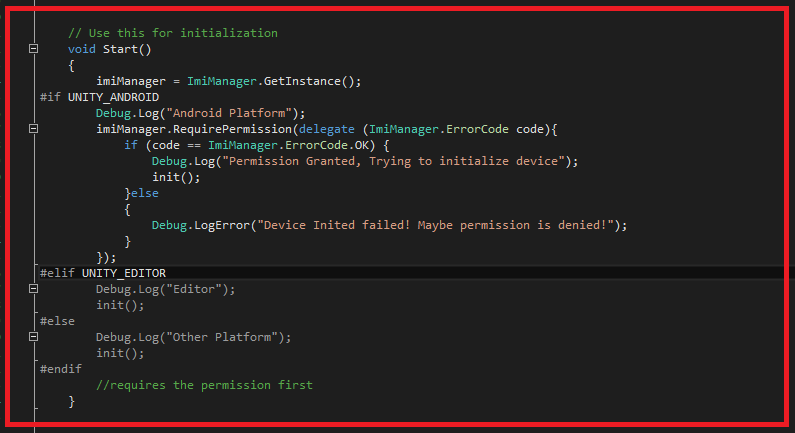


**Step 5:** Demo程序启动后，即可看到效果。

# 2. API使用说明：

**获取ImiManager的实例**

如果要自行开发，请先获取ImiManager的实例，如果是在Android平台，则额外申请获得权限:



如果所示，在Android平台上，获取权限需要传递一个回调函数，结果由参数传回。如果返回成功，可以说明已经获取到Usb读写权限，可以进行后续操作。否则无法调用摄像头。

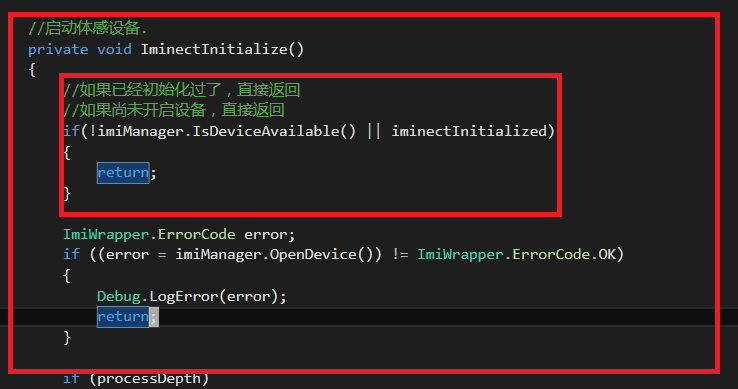
在其他平台上，获得ImiManager实例后，直接进行初始化即可。

**在主线程中调用初始化方法**

设备必须在主线程中被初始化。因此，你需要把Iminect/Prefabs/UnityMainThreadDispatcher 添加到场景中。这样才能保证获取Android权限的时候，返回的方法在主线程中被调用。

**打开设备**

要进行操作，首先打开设备



注意，在ImiWrapper中也有OpenDevice()方法和OpenDevice2()方法，但是ImiManager的OpenDevice()方法对前者进行了分装，根据获取权限的类别自动调用相应的方法，而且无需用户传递相应的参数。因此如果没有特殊理由，总应该先调用ImiManager的OpenDevice()方法。

另外，你总是可以通过ImiManager.IsDeviceAvailable()方法来判断设备是否获取了读写权限。

**其他操作**

其余相关的API，如打开流、读写和关闭流、关闭设备，请参考示例中的ImiDemo.cs文件，以及相关的注释。