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. 增加了多人骨骼的支持
6. 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
2. 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
3. 其他

1. 快速开始

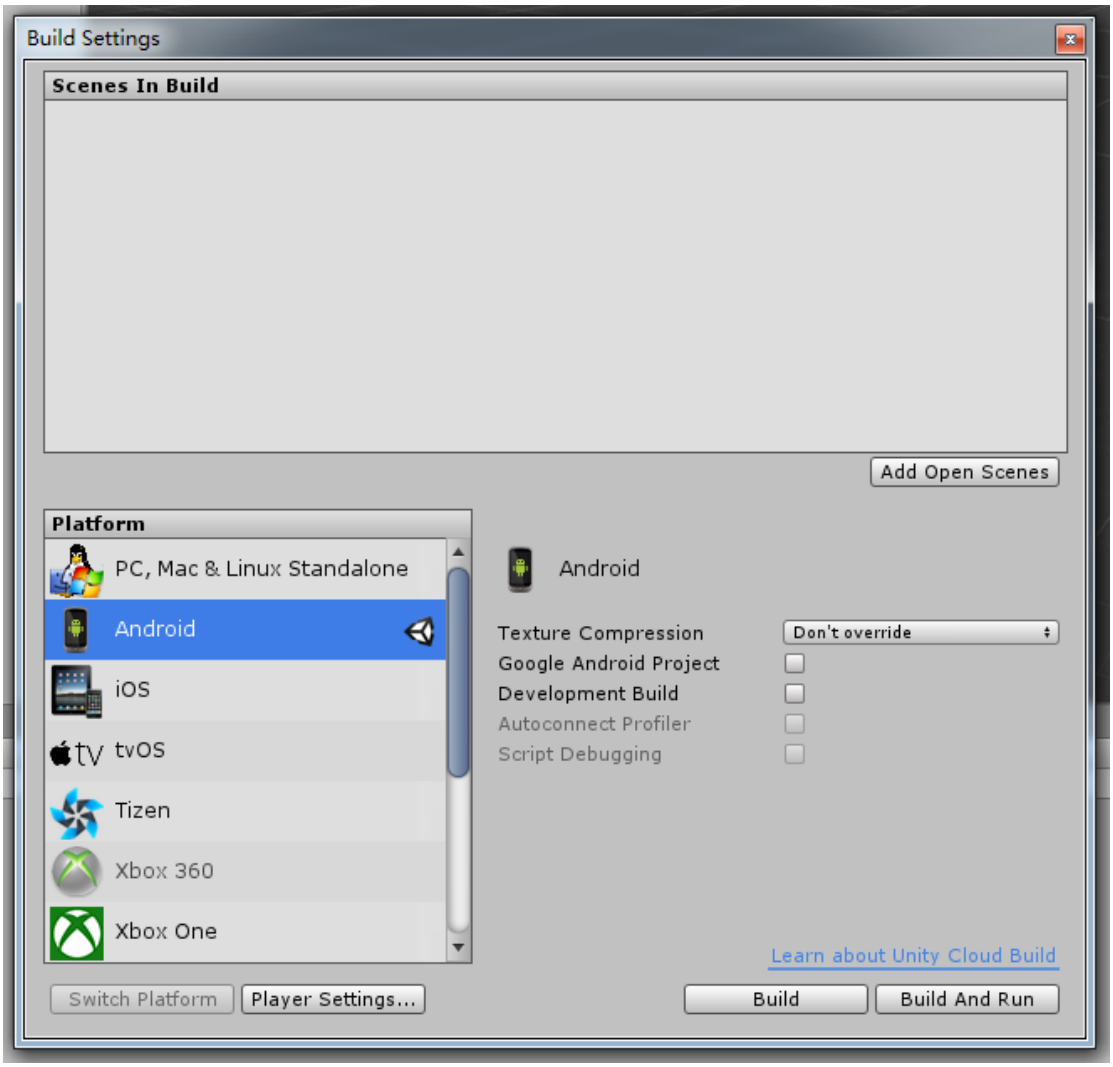
Step1 ： 将 imiUnitySDK_v0.1.3.unitypackage，任意导入到一个新的 Unity 工程中，导入全部资源

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

注意：如果你使用了之前最早的 SDK 版本，也请务必自己查看本部分，本版本修改了 Android 权限管理的逻辑，可以让游戏在没有 root 过的机型上顺利运行，并且不会出现发 release 版奔溃的 bug。

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:

```
    // Use this for initialization
    void Start()
    {
        imiManager = ImiManager.GetInstance();
#if UNITY_ANDROID
        Debug.Log("Android Platform");
        imiManager.RequirePermission(delegate (ImiManager.ErrorCode code){
            if (code == ImiManager.ErrorCode.OK) {
                Debug.Log("Permission Granted, Trying to initialize device");
                init();
            }else
            {
                Debug.LogError("Device Inited failed! Maybe permission is denied!");
            }
        });
#elif UNITY_EDITOR
        Debug.Log("Editor");
        init();
#else
        Debug.Log("Other Platform");
        init();
#endif
        //requires the permission first
    }
```

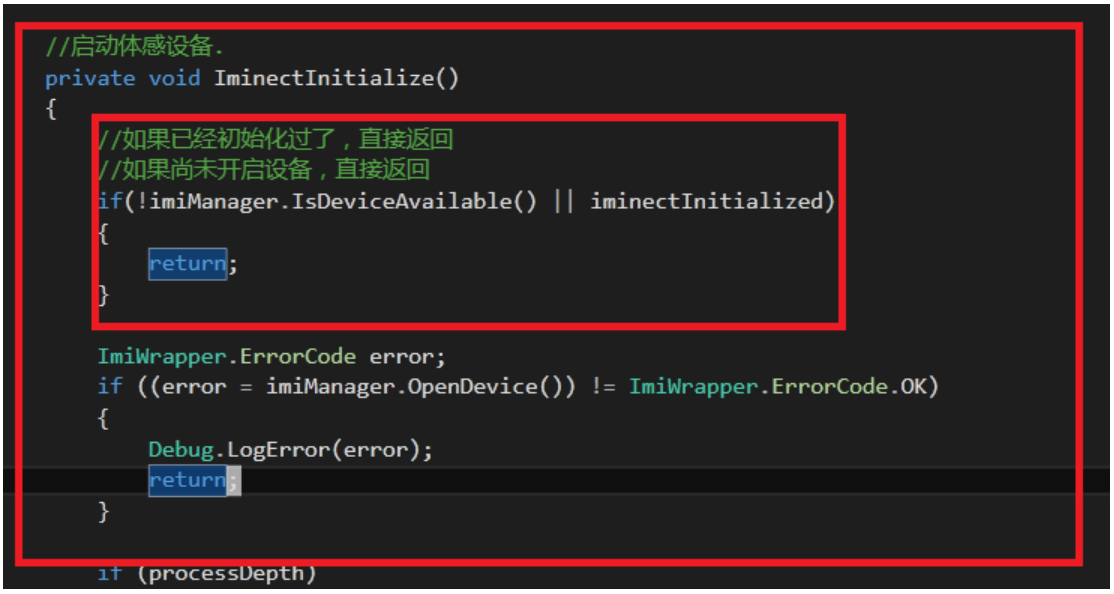
如果所示，在 Android 平台上，获取权限需要传递一个回调函数，结果由参数传回。如果返回成功，可以说明已经获取到 Usb 读写权限，可以进行后续操作。否则无法调用摄像头。
在其他平台上，获得 ImiManager 实例后，直接进行初始化即可。

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。

3. 常见问题

1. 新版的各个类都是干什么用的？

ImiManager.cs

这个类目前只提供了一个安卓上权限管理的功能。以后会作为一个帮助类，扩展功能并且提供一些方便操作的方法。

ImiDemo.cs

这个是一个实例类，基础的内容和之前版本的 IminectManager 类似，但是它的定位是帮助使用者了解如何使用基础的 ImiWrapper.cs 类

ImiWrapper.cs

这个类，可以认为是一个头文件，里面包含了一些数据结构的定义，以及摄像头最基础的接口。这个类所定义的方法均有 C++实现，通过这个类可以获取摄像头所提供的最原始数据，并且任由开发者调用和处理。

2. ImiDemo，ImiManager 与以前版本中的 IminectManager 有何关系？

其实是 IminectManager 被重新命名为 ImiDemo。因为之前 IminectManager 本来就是为了给开发者使用 ImiWrapper 提供实例写法，并非让开发者调用的。

ImiManager 是今后会不断扩充接口方便开发的主要类。ImiDemo 仍然是一个 ImiWrapper 调用示例。

如果你的程序已经调用了 IminectManager.cs 里面所提供的一些方法，没有关系，可以继续调用，但是以后 SDK 里面将不会继续提供 IminectManager。可以作为开发者自己项目中的脚本自行修改维护。