EasyAR 快速入门

本文档复制自 EasyAR 网站上一些基础入门指南。它将不会随着 EasyAR 网站内容变化而更新。
如果可能，请直接从网站上阅读文档：http://www.easyar.cn/view/documentapi.html

EasyAR 入门

EasyAR 是好用免费的全平台 AR（Augmented Reality，增强现实）引擎。

EasyAR 支持使用平面目标的 AR，支持 1000 个以上本地目标的流畅加载和识别，支持基于硬解码的视频（包括透明视频和流媒体）的播放，支持二维码识别。

EasyAR 支持 PC 和移动设备等多个平台，EasyAR 不会显示水印，也没有识别次数限制。

在拿到 EasyAR package 或 EasyAR 样例之后，你需要一个 key 才能使用。请确保在使用 EasyAR 之前阅读以下内容。

免费注册

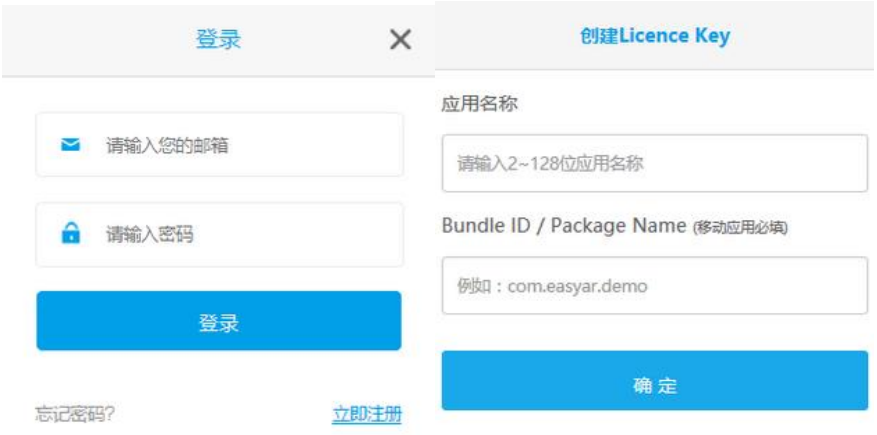
使用 EasyAR 之前需要使用邮箱在 www.easyar.cn 注册

**如果邮箱已经在视+官网（www.sightp.com）注册，可以直接登录。*

KEY 的获取

为了初始化 EasyAR SDK，需要先在网页系统中生成 key。

创建应用，输入应用名称和 Bundle ID（移动应用必填），点击确定后生成 Key



如果需要在 Android/iOS 设备上使用 EasyAR，必须填写 Bundle ID/Package Name，这个与你所创建的应用的 Bundle ID/Package Name 必须一致，否则可能初始化失败。

在创建应用之后，创建过程填写的数据仍可修改。

在创建之后，你可以在下图位置获得 key。

修改用户名（可选）

登录并点击右上角图标，编辑基本信息即可修改系统生成的用户名。EasyAR 网站和论坛使用相同的用户名，可以使用邮箱或用户名来登录 EasyAR 网站和论坛。你可以在 EasyAR 官网点击“社区”进入 EasyAR 论坛。

注意：用户名只可修改一次。

平台需求

平台支持

EasyAR 是跨平台的 AR SDK，支持以下操作系统

- Windows 7 及以上版本（**如果是在 Unity3D 中使用，支持 Windows 7 及以上版本，而独立的 Windows SDK 需要 Windows 7 SP1 及以上版本*）
- Mac OS X
- Android 4.0 及以上版本
- iOS 7.0 及以上版本

3D 引擎支持

- Android/iOS GLES2
- Unity 3D
- 未来更多（内嵌和外接）

Unity 兼容性

EasyAR 支持 Unity4（4.6 以上）和 Unity5。

Graphics API

- Windows: Direct3D9, Direct3D11, OpenGL2, OpenGLCore
- Mac OS X: OpenGL2, OpenGLCore
- Android: OpenGLES2
- iOS: OpenGLES2

编译运行 EasyAR 的 Unity 样例

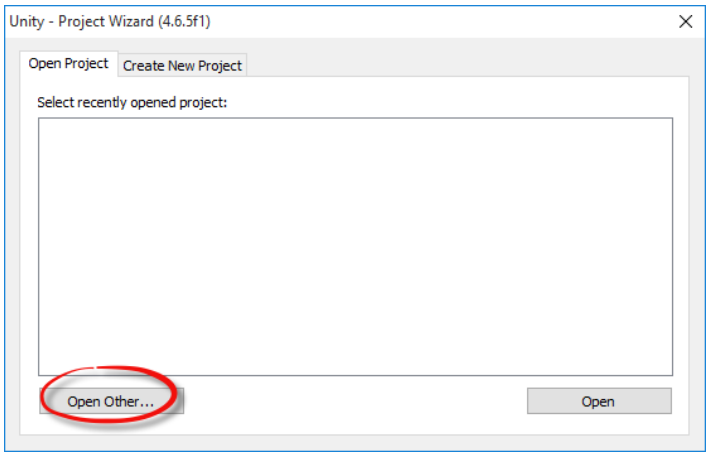
Pre-Requirements

- Unity 4.6 或更新版本
- （如果要发布 Android 应用）Android SDK with Build Tools 至少需要版本 20.0.0
- （如果要发布 iOS 应用）IPhone 或 IPad 或其它真实 Apple 设备（EasyAR 不支持在虚拟机上运行）

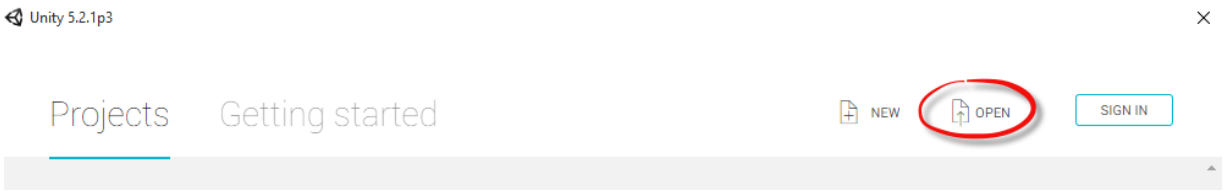
打开样例

首先你需要打开 unity 样例工程并打开样例中的场景。

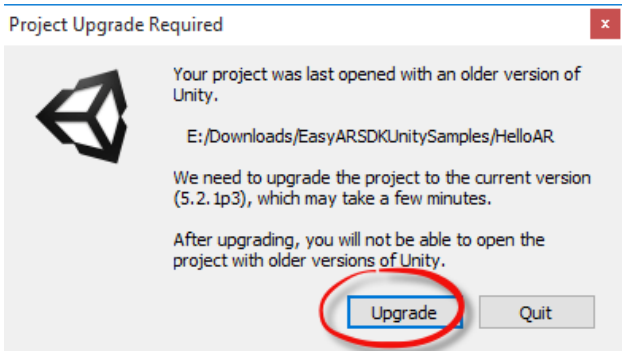
Unity 4 如下图所示，



而 Unity 5 如下，

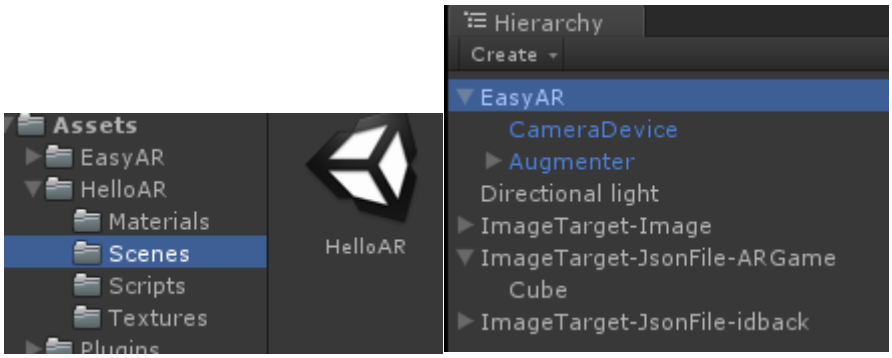


使用 Unity 5 打开工程之后，需要按照 Unity 的说明升级工程。升级之后就可以正常使用，配置和 Unity 4 相同。

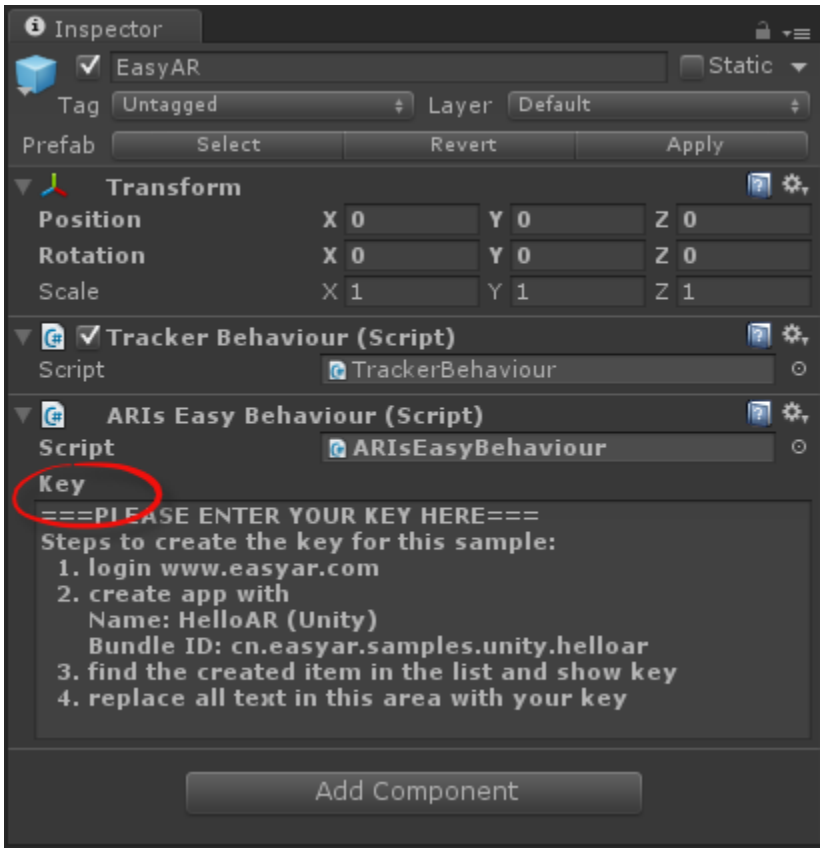


输入 Key

找到 ‘EasyAR’ object 并在 inspector 中输入‘Key’。如果 key 是非法的，程序初始化将会失败，可能显示黑屏。



按照下面的说明在 TextArea 中填入 Key。



就是这么简单！现在就可以在包括 Windows/Mac/Android/iOS 所有平台上运行 Unity 样例了。

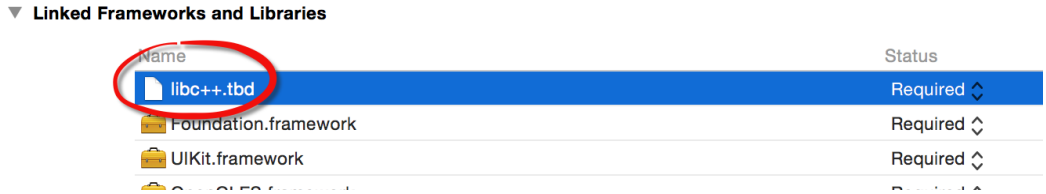
XCode 配置

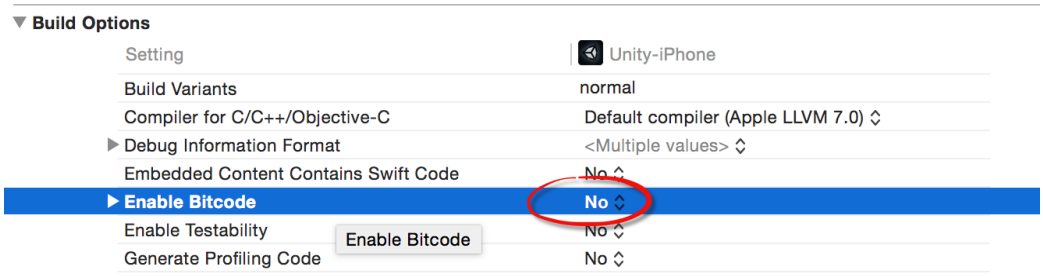
*如果你在使用最新的 Unity 版本，Unity 会自动完成这个步骤。

如果你在生成 iOS app，在 Unity 打包生成 Xcode 工程之后，需要多一步设置。

XCode 6.x: 添加 “libc++.dylib” 到链接选项中

XCode 7.x: 添加 “libc++.tbd” 到链接选项中。并设置 “Enable Bitcode” 为“NO”。





编译运行 EasyAR 的 Android 样例（非 Unity）

安装需求

- JDK 1.7 或更新版本
- Android Studio 1.5 或更新版本
- Android NDK r10e
- Android SDK with Build Tools 至少需要版本 20.0.0
- Android API 23 (可从 Android SDK Manager 中下载)
- **推荐安装最新版本的 NDK 和 SDK*

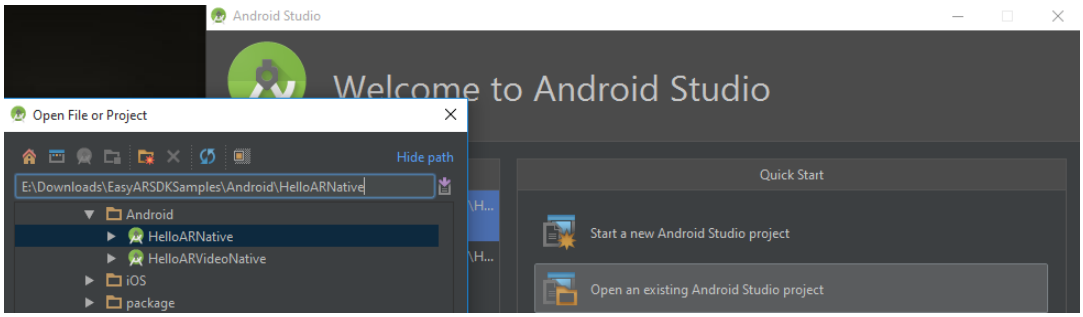
如果你是第一次使用 Android Studio，可以通过 Android 官方文档了解它
http://tools.android.com/tech-docs/new-build-system

对于 Android Studio 1.3 之后引入的 NDK 支持，可以参考
http://tools.android.com/tech-docs/android-ndk-preview

**根据官方的描述，Android Studio 对 NDK 的支持很可能在接下去的几个版本中会有变化。我们会跟踪这些改变，并在 Google 发布新版本后更新样例。我们可能不会更新随 SDK 一起发布的本文档。如果遇到最新版本 Android Studio 的问题，请到 EasyAR 网站查找最新样例代码。*

**请注意， EasyAR SDK 是支持在 Android Studio 1.4 和以下版本，以及 Eclipse 中引用和编译的。我们选择 Android Studio 1.5 作为样例创建工具的原因是，对于同时支持 Android Java 代码和 C++代码的集成配置和调试，它是目前最简单，设置最少的工具。*

在 Android Studio 中打开样例文件夹

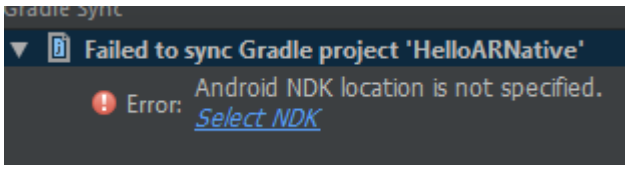


如果这是你第一次使用 Android Studio 应用于 Gradle 的实验插件，Android Studio 可能会需要一些时间来更新自身组件。

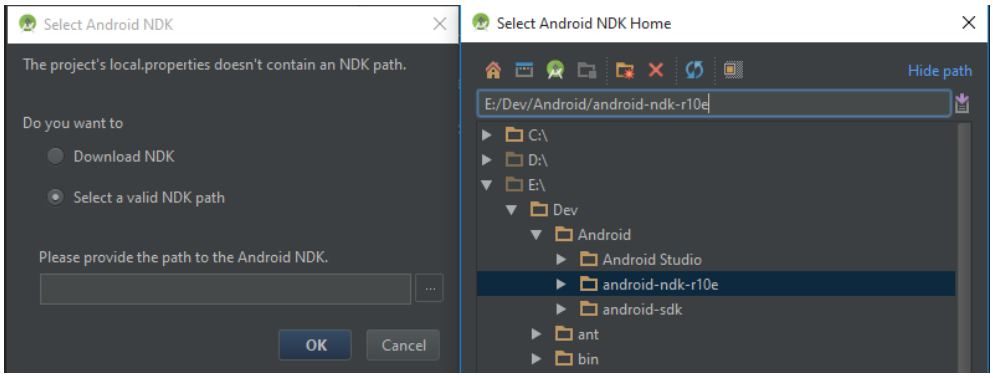
**你需要保持 “Android” 文件夹和“package” 文件存在，并保持它们的相对路径不变。*

设置 NDK 位置

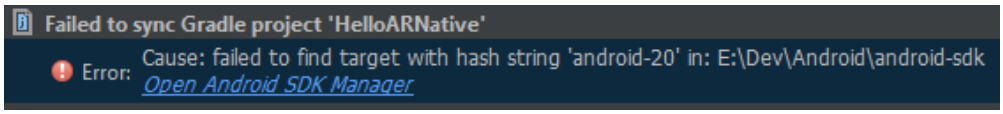
在打开工程之后，会看到如下错误信息。



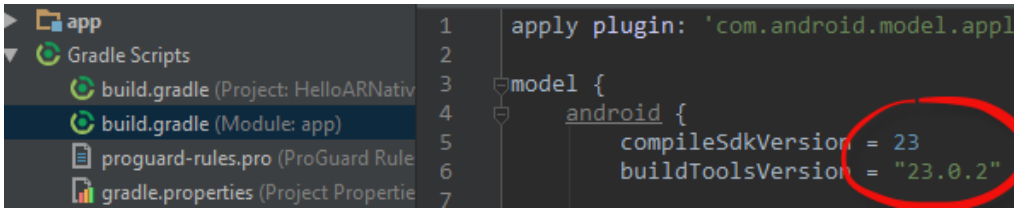
你可以点击错误信息中的链接使用如下方法设置本工程的 NDK 位置



如果你在使用与工程配置中不同的 Android API 或构建工具，可能会遇到如下错误信息

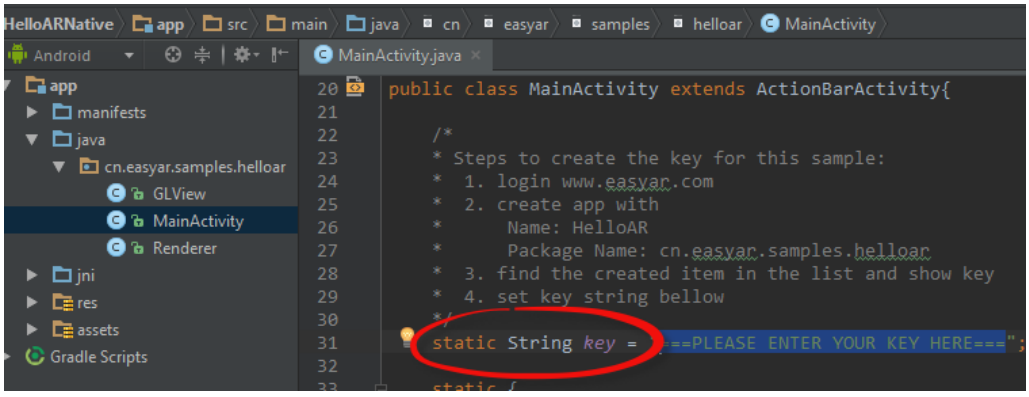


为了解决这个问题，可以在 Android SDK Manager 中按照错误信息中提供的版本或者修改 app 文件夹中的 build.gradle 文件，以匹配你现在所使用的版本。



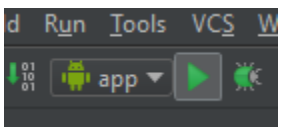
Key

根据下面说明设置 key。



运行

现在你就可以点击下图按钮，运行 Android 样例了。

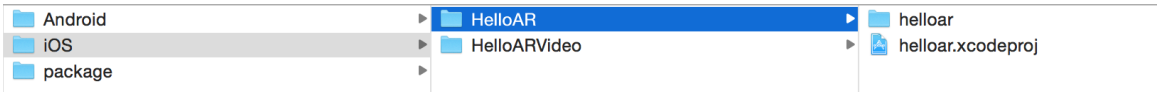


编译运行 EasyAR 的 iOS 样例（非 Unity）

安装需求

- XCode 6 或更新版本（我们在 XCode 6.4 和 XCode 7.1 中测试通过）
- IPhone 或 IPad 或其它真实 Apple 设备（EasyAR 不支持在虚拟机上运行）

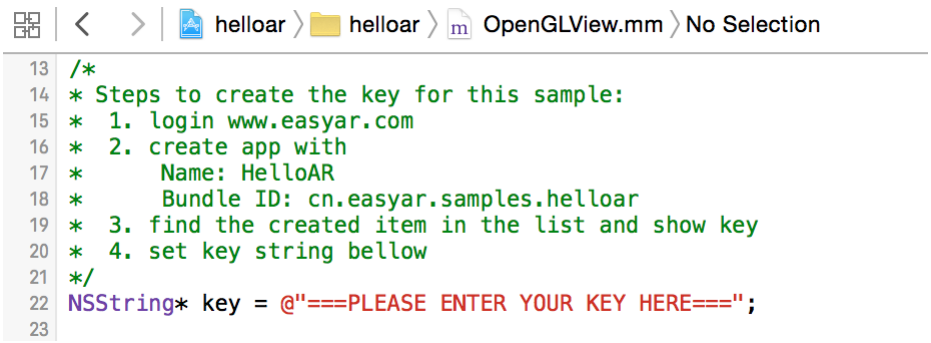
打开 XCode 工程



**你需要保持“iOS”文件夹和“package”文件存在，并保持它们的相对路径不变。*

Key

根据下面说明设置 key。



运行

现在你就可以点击下图按钮，运行 iOS 样例了。

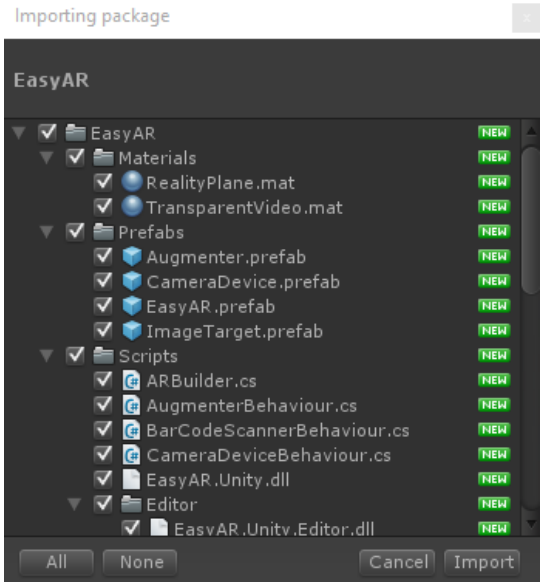
配置 EasyAR Unity SDK

安装需求

- Unity 4.6 或更新版本
- （如果要发布 Android 应用）Android SDK with Build Tools 至少需要版本 20.0.0
- （如果要发布 iOS 应用）IPhone 或 IPad 或其它真实 Apple 设备（EasyAR 不支持在虚拟机上运行）

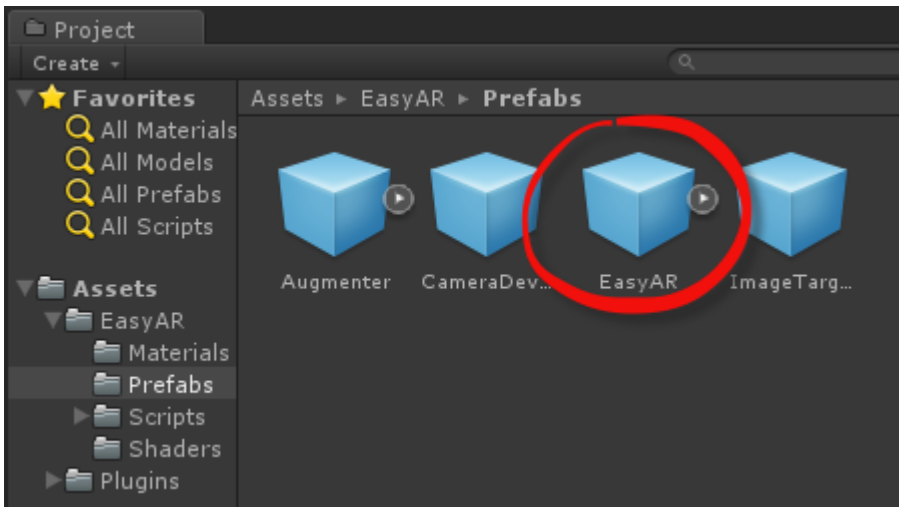
导入 Package

首先，你需要下载 EasyAR 压缩包，找到 EasyAR.unitypackage，打开并导入到 Unity 中。



初始化 EasyAR

为了让 EasyAR 正常工作，你需要将 EasyAR prefab 或其它 prefabs 添加到场景。拖拽 EasyAR Prefab 到场景中。



你可以在登录 EasyAR 网站后创建 key。然后使用这个 key 来初始化 EasyAR。可以添加如下两行代码到你的初始化代码中。

```
    ARBuilder.Instance.InitializeEasyAR(key);
    ARBuilder.Instance.EasyBuild();
```

如果使用默认配置（CameraDeviceBaseBehaviour.CaptureWhenStart 是启用的）EasyAR 将在 MonoBehaviour.Start 调用的时候开始运行。所以最好将上面两行代码放到 Awake 里面。

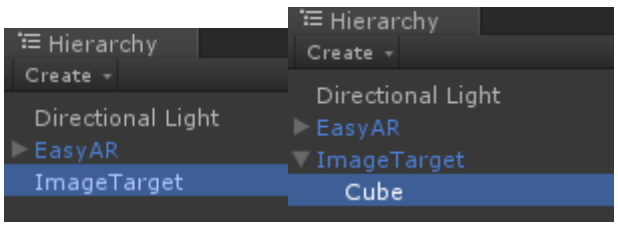
如果你想看到和样例一样的输入 key 的框，可以创建一个脚本并添加以下代码，并把这个脚本拖拽到 EasyAR prefab 上面。

```
using UnityEngine;
namespace EasyAR
{
    public class ARIsEasyBehaviour : MonoBehaviour
    {
        [TextArea(1, 10)]
        public string Key;
        private void Awake()
        {
            ARBuilder.Instance.InitializeEasyAR(Key);
            ARBuilder.Instance.EasyBuild();
        }
    }
}
```

添加 ImageTarget

ImageTarget 有许多用法，可以参考 HelloARTarget 样例。

如果需要在场景中静态配置 ImageTarget，需要拖拽一个 ImageTarget Prefab 到场景中。如何配置可以参考 ImageTarget Prefab 和 ImageTargetBaseBehaviour。



Target 事件

你可以在 ImageTargetBehaviour 中处理 target 相关事件

```
public class EasyImageTargetBehaviour : ImageTargetBehaviour, ITargetEventHandler
{
    void ITargetEventHandler.OnTargetFound(Target target)
    {
        Debug.Log("Found: " + target.Id);
    }
    void ITargetEventHandler.OnTargetLost(Target target)
    {
        Debug.Log("Lost: " + target.Id);
    }
    void ITargetEventHandler.OnTargetLoad(Target target, bool status)
    {
        Debug.Log("Load target (" + status + "): " + target.Id + " -> " + target.Name);
    }
    void ITargetEventHandler.OnTargetUnload(Target target, bool status)
```

```
    {
        Debug.Log("Unload target (" + status + "): " + target.Id + " -> " + target.Name);
    }
}
```

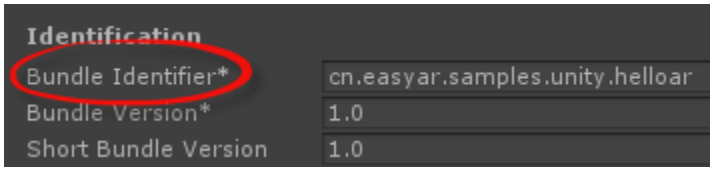
或是在实现了 ITargetEventHandler 接口的全局 target 管理器中处理 target 事件

```
public class EasyARTargetMananger : MonoBehaviour, ITargetEventHandler
{
    void ITargetEventHandler.OnTargetFound(Target target)
    {
        Debug.Log("Found: " + target.Id);
    }
    void ITargetEventHandler.OnTargetLost(Target target)
    {
        Debug.Log("Lost: " + target.Id);
    }
    void ITargetEventHandler.OnTargetLoad(Target target, bool status)
    {
        Debug.Log("Load target (" + status + "): " + target.Id + " -> " + target.Name);
    }
    void ITargetEventHandler.OnTargetUnload(Target target, bool status)
    {
        Debug.Log("Unload target (" + status + "): " + target.Id + " -> " + target.Name);
    }
}
```

你可以在这些事件中控制 ImageTarget 节点下物体的显示/隐藏。

Bundle ID（Android/iOS）

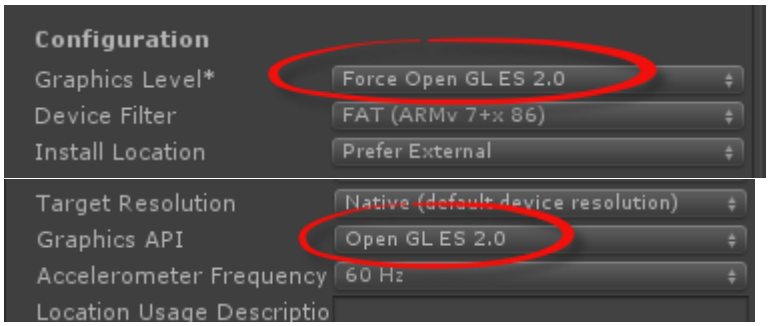
你需要在生成 Android/iOS 应用的时候设置 bundle ID。 bundle ID 应该与 easyar 网页上生成的 ID 相同。否则可能造成 SDK 初始化失败并黑屏。如果是在 Mac or Windows 上，这个 ID 就不需要了。



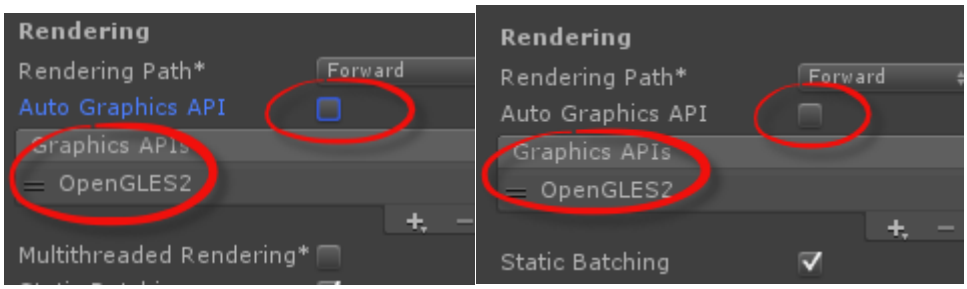
Graphics API（Android/iOS）

在导出 Android 和 iOS 应用的时候，需要设置 graphics API 为 OpenGL ES 2.0。这个设置在不同的 Unity 版本中有所不同。

Unity 4.x 中设置如下



Unity 5.x 设置如下



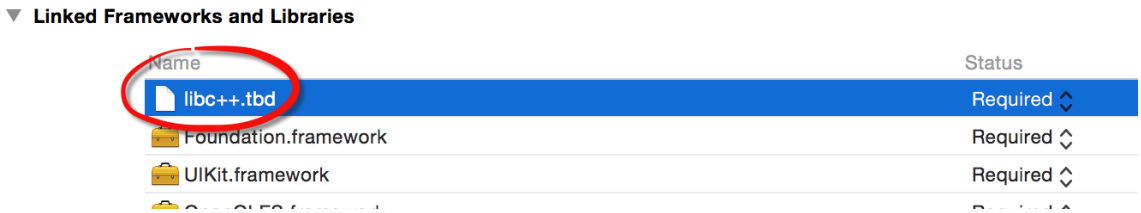
XCode 配置（iOS）

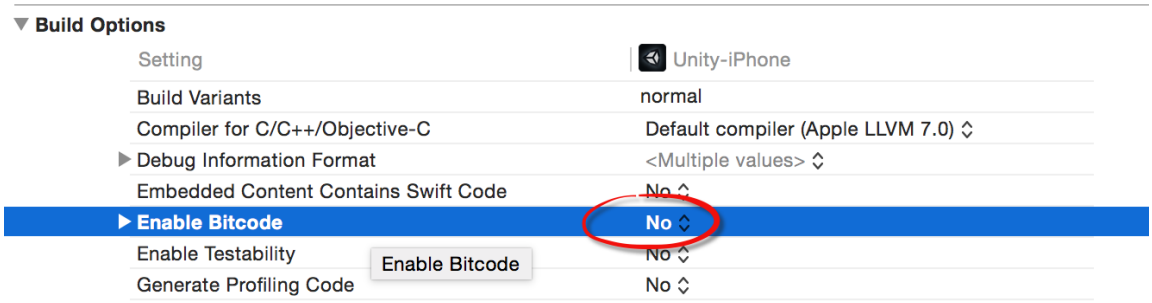
*如果你在使用最新的 Unity 版本，Unity 会自动完成这个步骤。

如果你在生成 iOS app，在 Unity 打包生成 Xcode 工程之后，需要多一步设置。

XCode 6.x: 添加 “libc++.dylib” 到链接选项中

XCode 7.x: 添加 “libc++.tbd” 到链接选项中。并设置 “Enable Bitcode” 为“NO”。





配置 EasyAR Android SDK（非 Unity）

安装需求

- JDK 1.7 或更新版本
- Android NDK
- Android SDK with Build Tools 至少需要版本 20.0.0
- **推荐安装最新版本的 NDK 和 SDK*

你可以在 Eclipse 或 Android Studio 中使用 EasyAR。我们建议和样例中一样使用 Android Studio 1.5，这样配置最为简单。

**注意：EasyAR 目前不支持纯 Java 的 API，你需要同时编写 Java 和 C++ 代码来使用 EasyAR 。可以参考样例中的实现。使用 EasyAR C++类是相对简单的，可以不需要担心指针和内存管理问题。我们会在今后的版本中加入这个接口。*

如果要像样例中一样在 Android Studio 1.5 中使用 EasyAR，你需要如下配置

- JDK 1.7 或更新版本
- Android Studio 1.5 或更新版本
- Android NDK r10e
- Android SDK with Build Tools 至少需要版本 20.0.0
- Android API 23 (可从 Android SDK Manager 中下载)

**注意：EasyAR SDK 是支持在 Android Studio 1.4 和以下版本，以及 Eclipse 中引用和编译的。我们选择 Android Studio 1.5 作为样例创建工具的原因是，对于同时支持 Android Java 代码和 C++ 代码的集成配置和调试，它是目前最简单，设置最少的工具。*

导入 EasyAR Android SDK

在 Eclipse 和 Android Studio 中导入 EasyAR 会有不同，而且在某些工具中你可能会需要写 Android.mk。这里我们会介绍在 Android Studio 1.5 中的配置细节。

首先你需要根据这篇官方文档修改 build.gradle。

在上面的修改之后，你可以添加 EasyAR 所需要的配置。

添加 EasyAR native 头文件路径

```
model {
    android.ndk {
        cppFlags.add("-I${file("/path/to/EasyARSDK/package/include")}".toString())
    }
}
```

你可能也会希望配置这些常用的内容

```
model {
    android.ndk {
        cppFlags.add("-DANDROID")
        cppFlags.add("-fexceptions")
        cppFlags.add("-frtti")
        stl = "gnustl_static"
```

```
        ldLibs.add("log")
        ldLibs.add("GLESv2")
    }
}
```

添加 EasyAR native library 依赖

```
model {
    android.sources {
        main {
            jni {
                dependencies {
                    library file("/path/to/EasyARSDK/package/Android/libs/armeabi-
v7a/libEasyAR.so") abi "armeabi-v7a"
                }
            }
        }
    }
}
```

添加 EasyAR Java library 依赖

```
dependencies {
    compile fileTree(include: ['*.jar'], dir: '/path/to/EasyARSDK/package/Android/libs')
}
```

最后你会得到一个类似于这样的 build.gradle

```
apply plugin: 'com.android.model.application'
model {
    android {
        compileSdkVersion = 23
        buildToolsVersion = "23.0.2"
        defaultConfig.with {
            applicationId = "cn.easyar.samples.helloar"
            minSdkVersion.apiLevel  =15
            targetSdkVersion.apiLevel = 22
            versionCode = 1
            versionName = "1.0"
        }
    }
    android.buildTypes {
        release {
            minifyEnabled = false
            proguardFiles.add(file("proguard-rules.pro"))
        }
    }
    android.ndk {
        moduleName = "HelloARNative"
        cppFlags.add("-I${file("../../../package/include")}".toString())
        cppFlags.add("-DANDROID")
        cppFlags.add("-fexceptions")
        cppFlags.add("-frtti")
        stl = "gnustl_static"
        ldLibs.add("log")
        ldLibs.add("GLESv2")
    }
    android.productFlavors {
        create("arm") {
            ndk.with {
                abiFilters.add("armeabi-v7a")
            }
        }
    }
    android.sources {
        main {
```

```
            jni {
                dependencies {
                    library file("../../../package/Android/libs/armeabi-
v7a/libEasyAR.so") abi "armeabi-v7a"
                }
            }
        }
    }
}
dependencies {
    compile fileTree(include: ['*.jar'], dir: '../../../package/Android/libs')
}
```

如果你在使用 Eclipse 或 Android Studio 1.4 及以下版本，你可能会需要写 Android.mk 来编译 C++代码。你可以参照上面的配置找到相关配置内容。

在 AndroidManifest 中添加权限

EasyAR 需要以下这些权限，缺少权限将会导致初始化失败并黑屏。

android.permission.CAMERA

android.permission.INTERNET

将这些添加到 AndroidManifest 中。

```
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
package="cn.easyar.samples.helloar" >
    <uses-permission android:name="android.permission.CAMERA" />
    <uses-permission android:name="android.permission.INTERNET" />
</manifest>
```

初始化 EasyAR

使用 EasyAR.initialize 来初始化 EasyAR。你可以添加初始化函数到你的 activity 中如下，

```
    protected void onCreate() {
        EasyAR.initialize(this, key);
    }
```

其它代码

剩下的就是写 EasyAR 的逻辑以及其它代码。EasyAR 同时需要 Java 和 C++代码。你可以参考 EasyAR 的样例来实现。

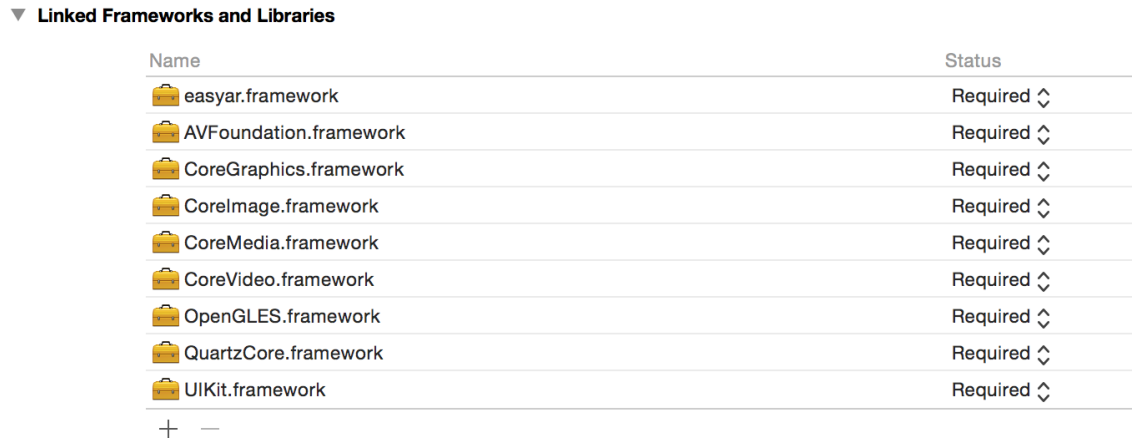
配置 EasyAR iOS SDK（非 Unity）

安装需求

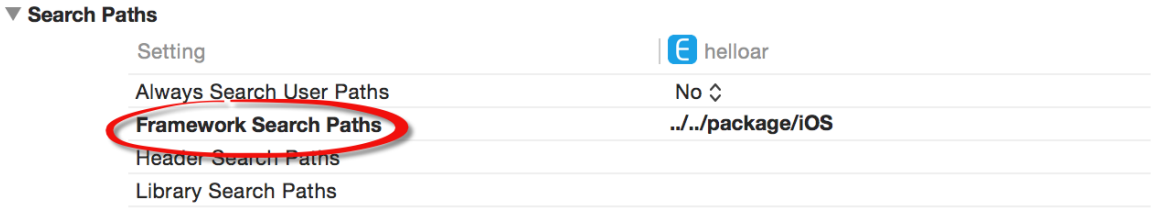
- XCode 6 或更新版本（我们在 XCode 6.4 和 XCode 7.1 中测试通过）
- IPhone 或 IPad 或其它真实 Apple 设备（EasyAR 不支持在虚拟机上运行）

添加 Framework

如果你在导入 EasyAR 创建自己的工程，你需要添加如下这些 framework。



对于 XCode 7.x，为了配置头文件引用路径，还需要手动设置 “Framework Search Paths”，让其包含 easyar.framework 的路径。



初始化 EasyAR

使用 EasyAR::initialize 来初始化 EasyAR。你可以添加初始化代码如下，

```
EasyAR::initialize([key UTF8String]);
```

设置 rotation

使用 EasyAR::setRotationIOS 来设置 rotation。

其它代码

剩下的就是写 EasyAR 的逻辑以及其它代码。你可以参考 EasyAR 的样例来实现。

配置 EasyAR Windows SDK（非 Unity）

安装需求

- Visual Studio 2015

初始化 EasyAR

使用 EasyAR::initialize 来初始化 EasyAR。

Augmenter

目前发布版本中 Augmenter API 被设为 NONE，也就是说目前还没有内置的 3D API。目前可以通过 Frame API 获取原始图像然后在自己创建的 GL/D3D/… 环境中进行渲染。目前除了这个 API 和视频播放以外 API 都可正常工作，与 Android/iOS 一致。我们会在今后的版本中逐渐添加这些缺失的特性。

你可以这样获取 frame 中的 image，

```
Frame frame = augmenter.newFrame(tracker);
Image iamge = frame.images()[0];
```

其它代码

剩下的就是写 EasyAR 的逻辑以及其它代码。你可以参考 EasyAR 的 Android 样例中的 C++代码来实现。大多数配置和使用方式都是和 Android native 代码是一样的，不同的是 Augmenter 的部分，这个在前面已有描述。

配置 EasyAR Mac SDK（非 Unity）

安装需求

- XCode 6 或更新版本（我们在 XCode 6.4 和 XCode 7.1 中测试通过）

初始化 EasyAR

使用 EasyAR::initialize 来初始化 EasyAR。

Augmenter

目前发布版本中 Augmenter API 被设为 NONE，也就是说目前还没有内置的 3D API。目前可以通过 Frame API 获取原始图像然后在自己创建的 GL/D3D/… 环境中进行渲染。目前除了这个 API 和视频播放以外 API 都可正常工作，与 Android/iOS 一致。我们会在今后的版本中逐渐添加这些缺失的特性。

你可以这样获取 frame 中的 image，

```
Frame frame = augmenter.newFrame(tracker);
Image iamge = frame.images()[0];
```

其它代码

剩下的就是写 EasyAR 的逻辑以及其它代码。你可以参考 EasyAR 的 Android 样例中的 C++代码来实现。大多数配置和使用方式都是和 Android native 代码是一样的，不同的是 Augmenter 的部分，这个在前面已有描述。