EasyAR Quick Start

This document is just a copy of some basic startup articles on EasyAR website. It will not reflect latest changes on the website. Please read on the web directly if possible.
http://www.easyar.com/view/documentapi.html

Getting Started with EasyAR

EasyAR is an Augmented Reality Engine. It is easy to use and free.

EasyAR supports AR based on planar target, supports smooth load and recognition for more than 1000 local targets, supports video playback based on HW codecs, supports transparent video and streaming video, supports QR code recognition.

EasyAR can be used in both PC and mobile platforms. EasyAR do not show watermarks, and have no limitation of recognition times.

When you have got EasyAR package or EasyAR samples, you would need a key. Make sure to read below steps before you start to use EasyAR.

Free Registration

Registration is required to use EasyAR.

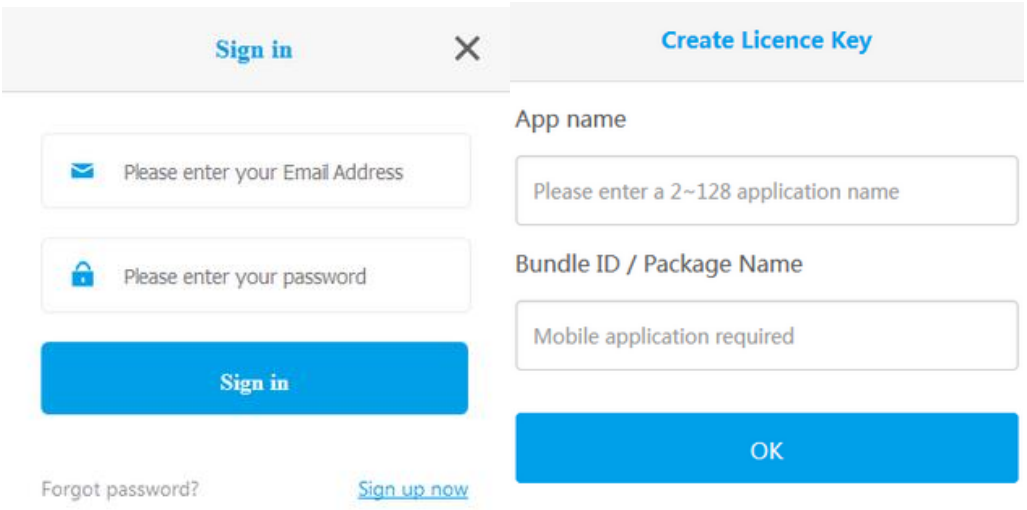
Register at www.easyar.com or www.easyar.cn using your email address.

**Alternatively if you have already registered at SightPlus website (www.sightp.com) using your email, you can directly login with that email address.*

Get a KEY

To initialize EasyAR SDK, a key is required.

Key is generated after you create an item after login.



Bundle ID/Package Name is required if you are creating mobile apps.

You can modify the values after the creation. You will get a key in the following place

Application name	Status	Date	License Key	Bundle ID/Package Name	Delete
HelloAR	Active	2015.12.22	Show	cn.easyar.samples.helloar	

Change User Name (Optional)

Login and click the icon in the right corner, you can edit user name in the next page. Currently this page and the forum are displayed in Chinese. User name is displayed at the right corner as an identifier in the forum. You can login EasyAR website and forum use either mailbox or username.

Note: You can change the user name only once.

Platform Requirements

Platform Support

EasyAR is a cross-platform AR SDK. The following OSes are supported

- Windows 7 and above (** support Windows 7 and above if used with Unity3D, the standalone Windows SDK requires Windows 7 SP1 and above*)
- Mac OS X
- Android 4.0 and above
- iOS 7.0 and above

3D Engine Support

- Android/iOS GLES2
- Unity 3D
- More in future (build-in and extern)

Unity Compatibility

Both Unity 4 (4.6+) and Unity 5 are supported.

Graphics API

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

Compile and Run EasyAR Unity Samples

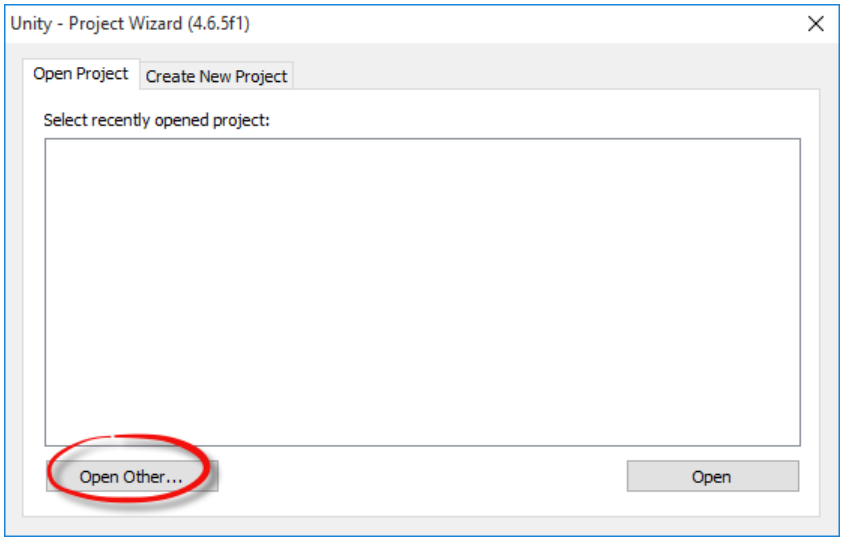
Pre-Requirements

- Unity 4.6 or later
- (If target for Android) Android SDK with Build Tools at least version 20.0.0
- (If target for iOS) IPhone or IPad device, or other real Apple devices (EasyAR do not support running on the simulator)

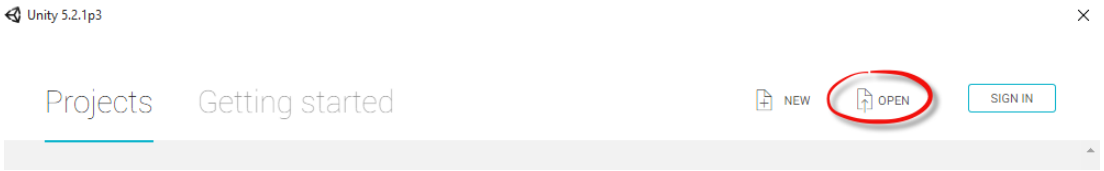
Open Sample

First you need to open unity sample project and open the scene in the sample.

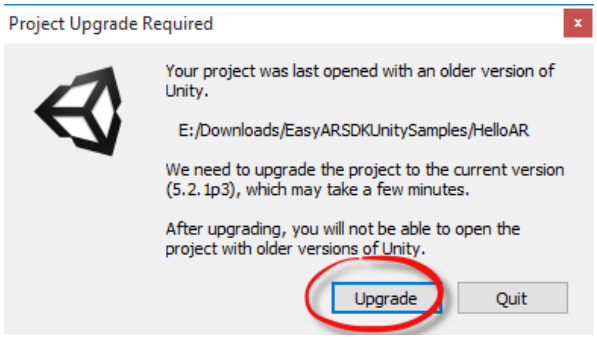
Unity 4 is like bellow,



And Unity 5 like this,

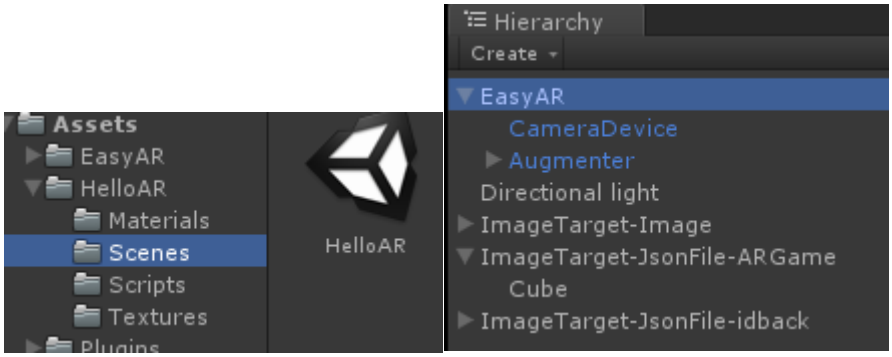


Additional for Unity 5, after open the project, follow Unity instructions to upgrade the project, then you can use it with no difference compared to Unity 4.

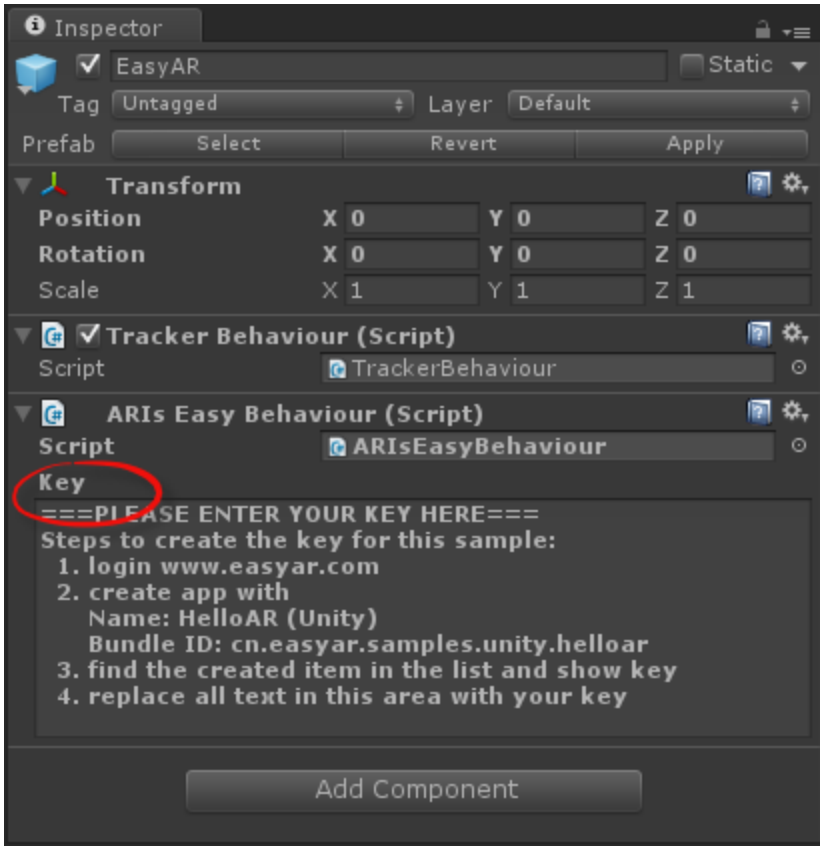


Enter Key

Find "EasyAR" object and Enter "Key" in the inspector. Initialize will fail if a valid key is not provided.



Follow the instructions in the TextArea to fill the key.



All is done! You can now run the Unity sample on all platforms including Windows/Mac/Android/iOS.

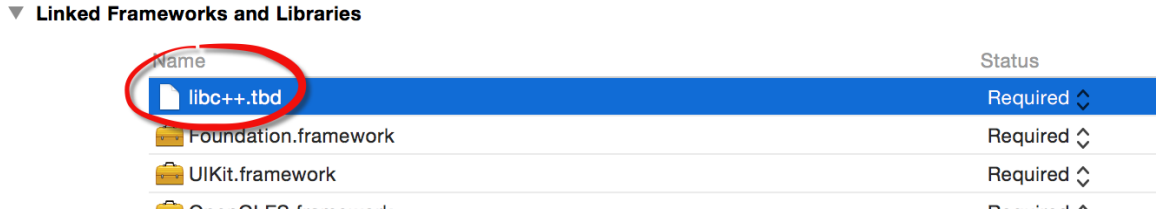
XCode configure

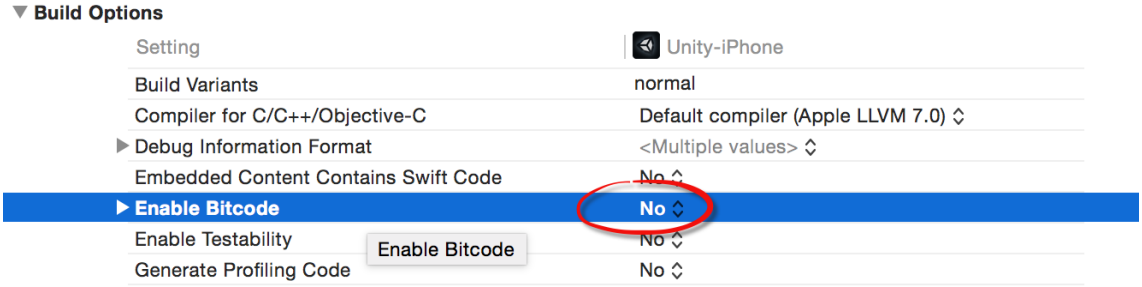
**If you are using latest Unity, this step is automatically done by Unity.*

When you are generating iOS apps, after the automatic build step which put everything from Unity to XCode, you need one more step to make all things work.

XCode 6.x: add *libc++.dylib* into linker libraries.

XCode 7.x: add *libc++.tbd* into linker libraries. And Set *Enable Bitcode* to *NO*.





Compile and Run EasyAR Android Samples

Pre-Requirements

- JDK 1.7 or later
- Android Studio 1.5 or later
- Android NDK r10e
- Android SDK with Build Tools at least version 20.0.0
- Android API 23 (download from Android SDK Manager)
- **It is recommended to install the latest version of NDK and SDK*

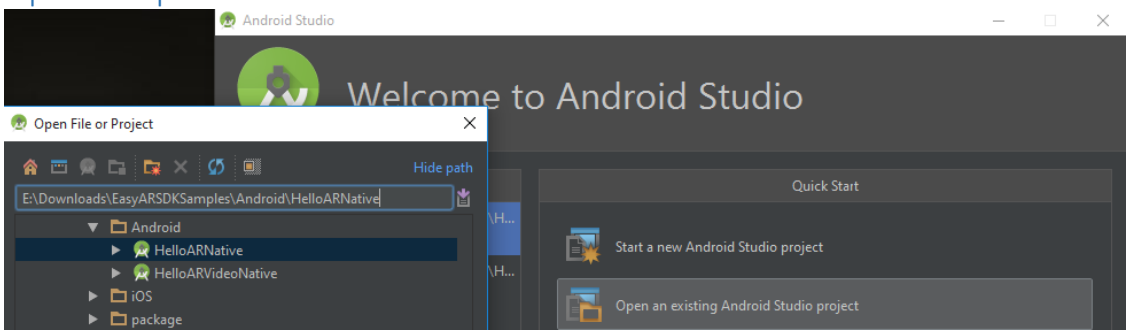
If you are new to Android Studio, you can read the official doc for reference.
http://tools.android.com/tech-docs/new-build-system

For the newly introduced NDK support since Android Studio 1.3, you can reference
http://tools.android.com/tech-docs/android-ndk-preview

**According to the official description, it is highly possible that the NDK support will change across the next a few releases. We will follow-up the changes and provide samples using latest Android Studio when Google release a new stable version. We may not update this doc in SDK package, please always check EasyAR website for latest sample.*

**Please note EasyAR SDK do support building from Android Studio 1.4 or below, or from Eclipse. We choose Android Studio 1.5 for sample creation because it is the best tool for now that offers a simple way to configure and debug Android Java code and C++ code together.*

Open sample folder from Android Studio

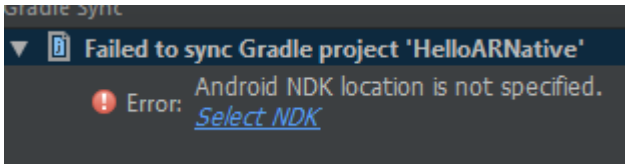


If it is the first time you are using the new Android Studio experimental plugin for Gradle, Android Studio may need some time to update its components.

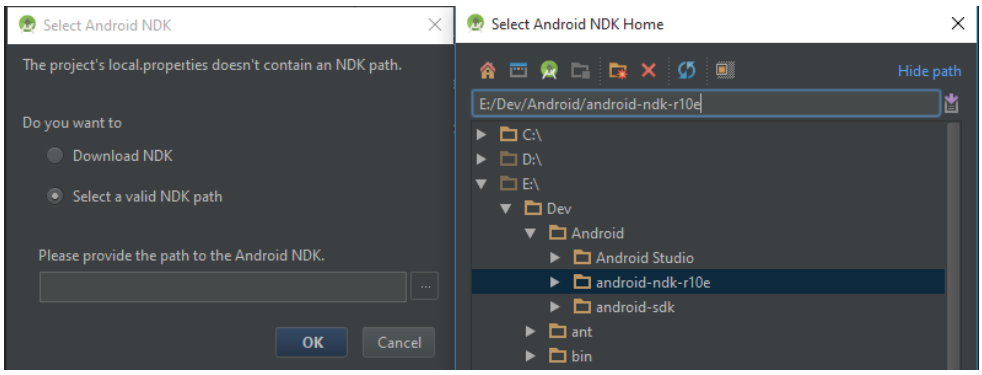
** You need to keep "Android" folder and "package" folder exist and keep their path relatively unchanged to build and run Android samples.*

Set NDK location

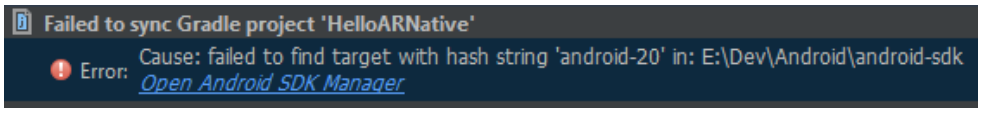
After open the project, you will see an error message like this



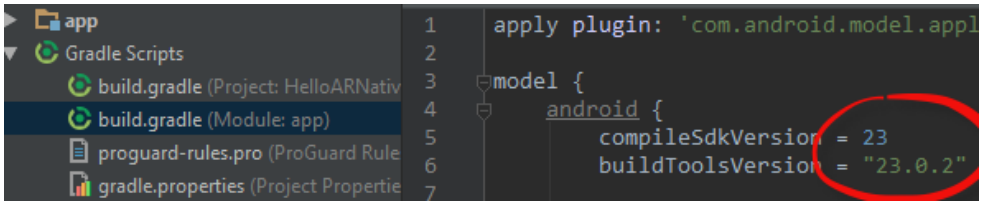
You can click the link in error message and set NDK location to this project like below



If you are using an Android API or build tool different from the project setting, you may get an error like this

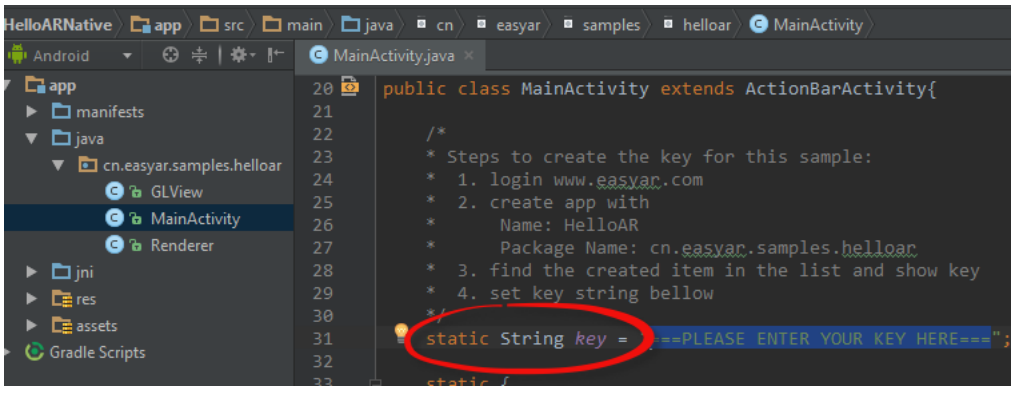


To solve this, you can install the version proposed by the error message from Android SDK Manager or change build.gradle strings in the app folder below to match your version.



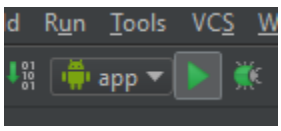
Key

Follow below instructions to set the key.



Run

Now you can run the sample by clicking the following button

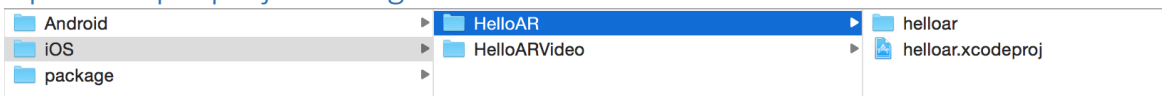


Compile and Run EasyAR iOS Samples

Pre-Requirements

- XCode 6 or later (we have tested in XCode 6.4 and XCode 7.1)
- IPhone or IPad device, or other real Apple devices (EasyAR do not support running on the simulator)

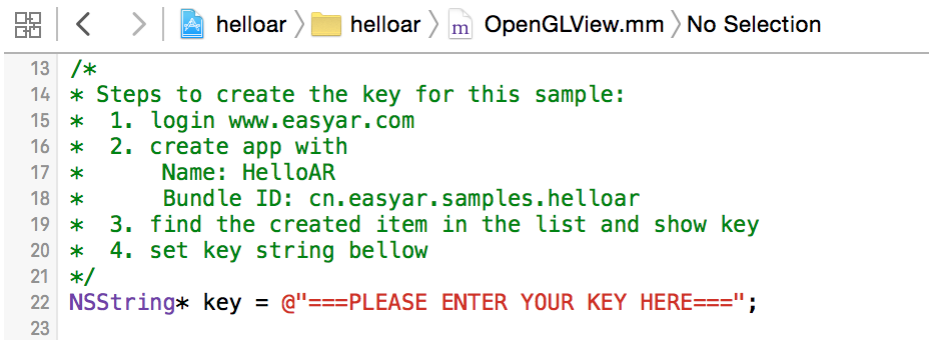
Open sample project using XCode



** You need to keep "iOS" folder and "package" folder exist and keep their path relatively unchanged to build and run iOS samples.*

Key

Follow below instructions to set the key.



Run

Now you can run the sample by clicking the following button

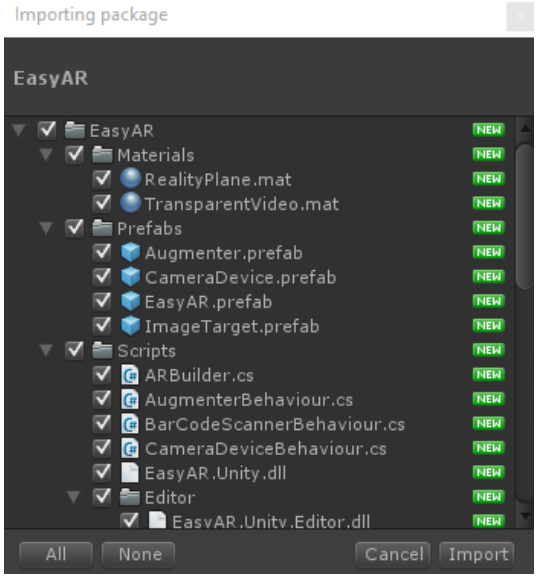
Setting up EasyAR Unity SDK

Pre-Requirements

- Unity 4.6 or later
- (If target for Android) Android SDK with Build Tools at least version 20.0.0
- (If target for iOS) IPhone or IPad device, or other real Apple devices (EasyAR do not support running on the simulator)

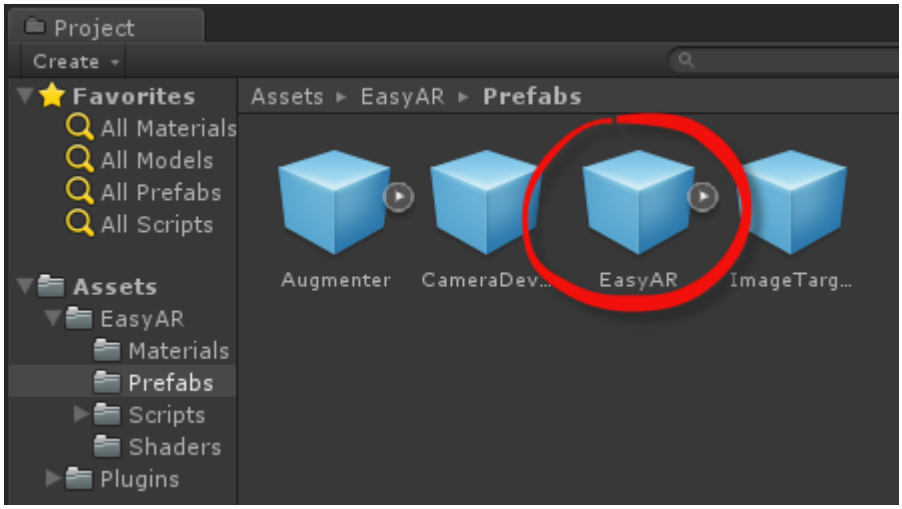
Import Package

First, you have to download EasyAR package, find EasyAR.unitypackage and open to import it into Unity.



Initialize EasyAR

To make EasyAR work, you have to have EasyAR prefab or other prefabs in the scene. Drag EasyAR Prefab into the scene.



You can create a key after login EasyAR website. Next you need to add two lines into your initialize code to initialize EasyAR with your key.

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

If you use the default configure (CameraDeviceBaseBehaviour.CaptureWhenStart is enabled), EasyAR will begine to run at MonoBehaviour.Start. So it is better to put above code into Awake.

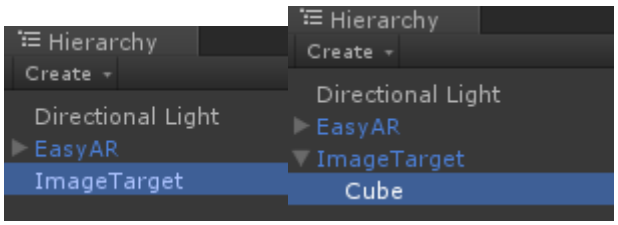
If you want to see the same input text area like the samples, you can create a script and write the following code. Then drag the script onto the EasyAR prefab.

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

Add ImageTarget

There are many ways to use the ImageTarget. You can reference the HelloARTarget sample.

If you want to setup ImageTarget statically in the scene, you have to drag an ImageTarget Prefab into the scene. Read ImageTarget Prefab and ImageTargetBaseBehaviour for details about configurations.



Target Events

You can handle target event either in ImageTargetBehaviour

```
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);
    }
}
```

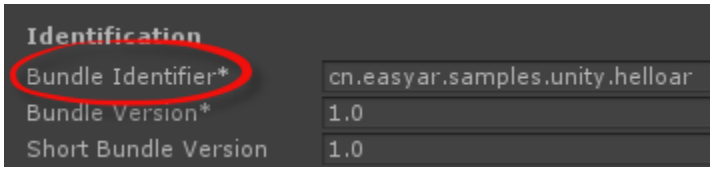
or in a global target manager that implements ITargetEventHandler

```
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);
    }
}
```

You can show/hide objects under ImageTarget in the target events.

Bundle ID (Android/iOS)

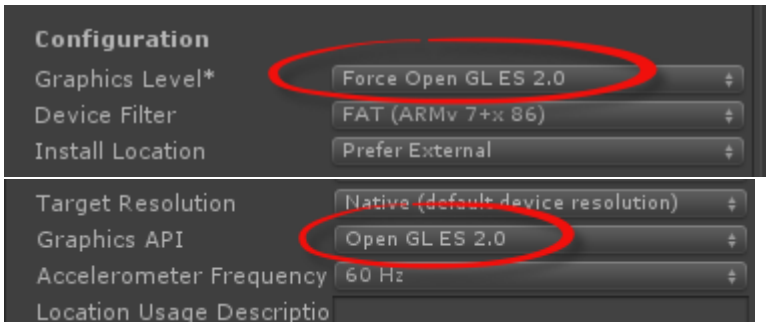
You need to set bundle ID when generating Android/iOS apps. The bundle ID should match the ID where the key is generated in the EasyAR website. Otherwise the initialize will fail. One exception is for Mac or Windows; such ID matches are not required on these two platforms.



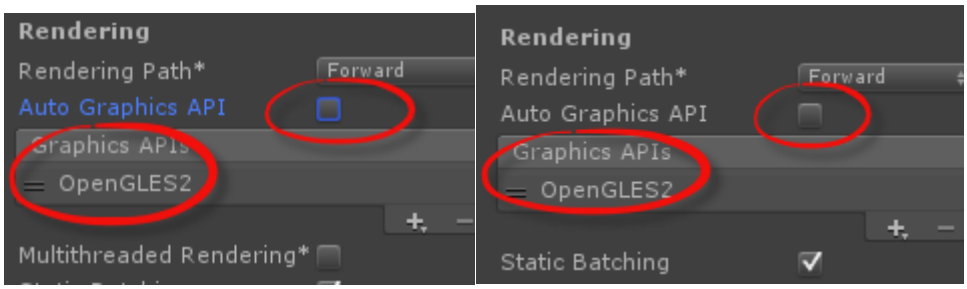
Graphics API (Android/iOS)

Set graphics API to OpenGL ES 2.0 if you are building Android or iOS apps. This setting is different across Unity version.

Unity 4.x settings are like this



Unity 5.x settings are



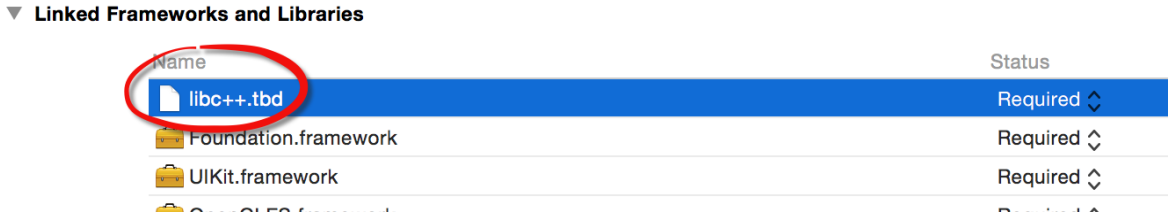
XCode configure (iOS)

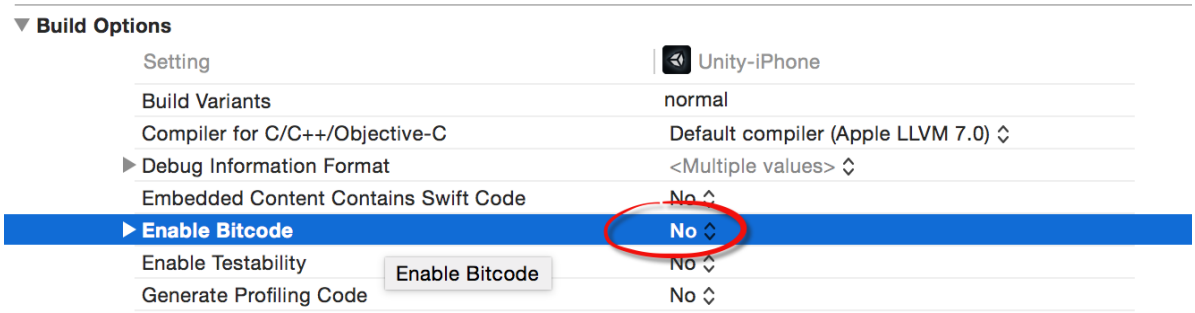
**If you are using latest Unity, this step is automatically done by Unity.*

When you are generating iOS apps, after the automatic build step which put everything from Unity to XCode, you need one more step to make all things work.

XCode 6.x: add *libc++.dylib* into linker libraries.

XCode 7.x: add *libc++.tbd* into linker libraries. And Set *Enable Bitcode* to *NO*.





Setting up EasyAR Android SDK

Pre-Requirements

- JDK 1.7 or later
- Android NDK
- Android SDK with Build Tools at least version 20.0.0
- **It is recommended to install the latest version of NDK and SDK*

You can use EasyAR in Eclipse or Android Studio. For simple configuration, we suggest to use Android Studio 1.5 as we did in samples.

**Note: EasyAR do not support Java only API for now, you have to program both Java code and C++ code to make EasyAR work. You can reference the samples to do so. It is relatively simple to use EasyAR C++ classes, you do not have to worry about pointers and memory management. We will add Java API support in future releases.*

To use EasyAR in Android Studio 1.5 like the samples, you may also need the followings

- JDK 1.7 or later
- Android Studio 1.5 or later
- Android NDK r10e
- Android SDK with Build Tools at least version 20.0.0
- Android API 23 (download from Android SDK Manager)

**Note: EasyAR SDK do support building from Android Studio 1.4 or below, or from Eclipse. We choose Android Studio 1.5 for sample creation because it is the best tool for now that offers a simple way to configure and debug Android Java code and C++ code together.*

Import EasyAR Android SDK

This step is different in Eclipse and Android Studio, and you may need to write Android.mk in some tools. Here we will introduce the configuration details when using Android Studio 1.5.

First you have to change your build.gradle according to this official article.

After above change, you can now add EasyAR specific configurations

Add EasyAR native include directories

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

You may also want to add some common native configurations

```
model {
    android.ndk {
        cppFlags.add("-DANDROID")
        cppFlags.add("-fexceptions")
        cppFlags.add("-frtti")
        stl = "gnustl_static"
        ldLibs.add("log")
        ldLibs.add("GLESv2")
```

```
    }
}
```

Add EasyAR native library dependencies

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

Add EasyAR Java library dependencies

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

Finially you may have a build.gradle like this

```
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')
}
```

If you are using Eclipse or Android Studio 1.4 or bellow, you may need to write Android.mk. You can find relative settings like above.

Add Permissions in AndroidManifest

EasyAR require the following permissions, missing permissions may cause initialize fail.

android.permission.CAMERA

android.permission.INTERNET

```
<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>
```

Initialize EasyAR

Use EasyAR.initialize to initialize EasyAR. You can add the initialize into your activity like this.

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

Other Code

The reset is to write EasyAR logics and other code. Both Java code and C++ code are needed. You can reference EasyAR samples for more details.

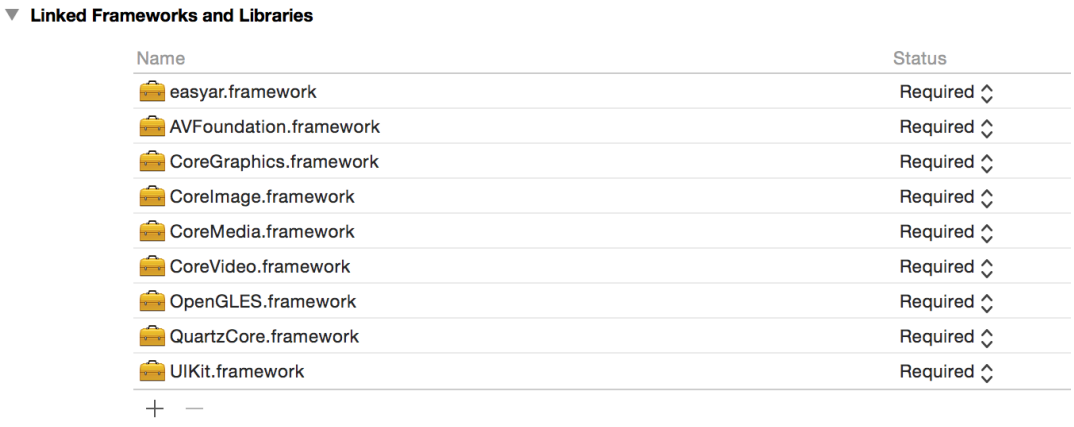
Setting up EasyAR iOS SDK

Pre-Requirements

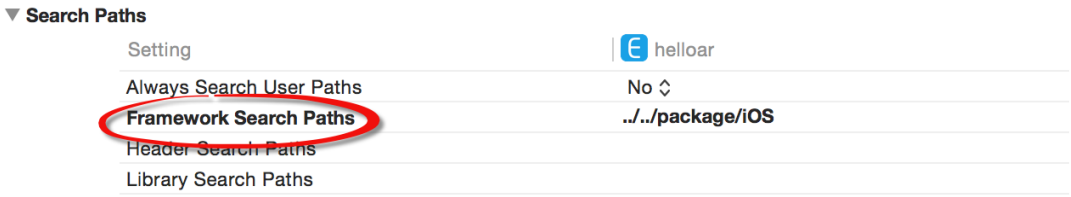
- XCode 6 or later (we have tested in XCode 6.4 and XCode 7.1)
- IPhone or IPad device, or other real Apple devices (EasyAR do not support running on the simulator)

Add Frameworks

If you are creating your own project directly from EasyAR package, you need to add these frameworks.



And for XCode 7.x, you need to manually set *Framework Search Paths* to include the path of easyar.framework. This is needed to include EasyAR headers from framework.



Initialize EasyAR

Use EasyAR::initialize to initialize EasyAR. You can add the initialize into your code like this.

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

Set rotation

Use EasyAR::setRotationIOS to set rotation.

Other Code

The reset is to write EasyAR logics and other code. You can reference EasyAR samples for more details.

Setting up EasyAR Windows SDK

Pre-Requirements

- Visual Studio 2015

Initialize EasyAR

Use EasyAR::initialize to initialize EasyAR.

Augmenter

Currently the Augmenter API is set to NONE in the release, which means no explicit 3D API is built in. Alternatively, you can get raw image directly from Frame API and do drawings in GL/D3D/… environment setup outside the SDK. All other APIs except video playback work the same as Android/iOS. Those missing features will be added in future releases.

You can get the image from frame like this.

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

Other Code

The reset is to write EasyAR logics and other code. You can reference C++ code in EasyAR Android samples for more details. Most configurations and usages are same with Android native code except the Augmenter described above.

Setting up EasyAR Mac SDK

Pre-Requirements

- XCode 6 or later (we have tested in XCode 6.4 and XCode 7.1)

Initialize EasyAR

Use EasyAR::initialize to initialize EasyAR.

Augmenter

Currently the Augmenter API is set to NONE in the release, which means no explicit 3D API is built in. Alternatively, you can get raw image directly from Frame API and do drawings in GL/D3D/… environment setup outside the SDK. All other APIs except video playback work the same as Android/iOS. Those missing features will be added in future releases.

You can get the image from frame like this.

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

Other Code

The reset is to write EasyAR logics and other code. You can reference C++ code in EasyAR Android samples for more details. Most configurations and usages are same with Android native code except the Augmenter described above.