# How to Build and Update an HLS Streaming AR App in Unity

Unity 2020.1.11f1 (free)
**EasyAR** (personal: free / pro: $468 annually)
AVProVideo ($150 per platform)
DOTWeenPro ($15)

The goal: build an AR app that, when an image is triggered, plays a video that is hosted externally, and pauses the video when the image is no longer tracked. Note: I would recommend Vuforia over EasyAR as it lags on iOS, but if you have to use EasyAR and are having trouble building, this template can be a good jumping off point.

## TABLE OF CONTENTS

# 1.0 Preparing your Media for Streaming

**(Any videos that you want to play when a trigger is activated in AR)**

Hello! Okay, we'll need to convert your .mv4 video file into something called an .m3u8 Master playlist

Basically what that is is a Playlist and subfolders with several chunks of video and audio inside each one that will load one at a time and streamed seamlessly. It needs to be broken down in chunks so it's not having to prep and load a massive video all at once, if it does it in smaller sizes it can stream more rapidly without a need to buffer for a long time.

Each folder is a different quality. The m3u8 file wont switch between qualities. It's just a playlist file, which contains a list of references to different videos. It's the responsibility of whatever is executing the code (in this case, the plugin you're using) to choose between videos.

There are two ways to convert videos into .m3u8 playlists: you can do it yourself, or pay like, 5 cents to have a website do it for you. I recommend not doing this yourself!!!!

## 1.1 Converting videos yourself: install homebrew and ffmpeg

**I DON'T RECOMMEND THIS MY DUDES**
This is a ...skill. It takes a LOT of time to come up with the right "recipe" to suit your needs, and to figure out the right bitrate and error margin, and what you know a clients network is set up for, but if you want to do it yourself here's what you can do (only know how to do this on mac:
on a mac, in terminal:

- Terminal, install homebrew - can copy paste this in:
  - /usr/bin/ruby -e "$(curl -fsSL https://raw.githubusercontent.com/Homebrew/install/master/install)"
  - brew install ffmpeg

## 1.1.2 Have ffmpeg convert .mp4 file to packets + .m3u8 playlist:

**basic example (do this in terminal):**

- ffmpeg -i /Users/[yourusername]/Documents/01-FalconHeavy/FHVid.mp4 -profile:v baseline -level 3.0 -s 640x360 -start_number 0 -hls_time 10 -hls_list_size 0 -f hls FHVid.m3u8

**Heres an example that I eventually came up with:**

- .0390625ffmpeg -i /Users/[yourusername]/Documents/01-FalconHeavy/FHVid.mp4 \
  -b:v:0 5000k -maxrate 5350k -bufsize 7500k \
  -b:v:1 2800k -maxrate 2996k -bufsize 4200k \
  -b:v:2 1400k -maxrate 1498k -bufsize 2100k \
  -b:v:3 800k -maxrate 856k -bufsize 1200k \
  -b:a:0 192k \
  -b:a:1 128k \
  -b:a:2 128k \
  -b:a:3 96k \
  -c:a aac -ar 48000 -c:v h264 -profile:v main -crf 20 -sc_threshold 0 -g 48 -keyint_min 48 -map 0:v -map 0:a -map 0:v -map 0:a -map 0:v -map 0:a -map 0:v -map 0:a \
  -f hls -var_stream_map "v:0,a:0  v:1,a:1 v:2,a:2 v:3,a:3" \
  -master_pl_name FHVidmaster.m3u8 -hls_time 4 -hls_playlist_type vod \
  -hls_segment_filename '/Users/[yourusername]/Documents/01-FalconHeavy/01-FalconHeavy/HLS/file_%v_%03d.ts' /Users/[yourusername]/Documents/01-FalconHeavy/HLS/out_%v.m3u8 \

## 1.2 Do it quick and online for like... less than 50 cents

**https://cloud.qencode.com**
The support team there are nice too, lol. They've been helpful to me a lot!

- make an account
- you can select what types of video qualities you want too.
- way quicker and faster and more efficient than doing it yourself.

## 1.3 Storing the files in the cloud for streaming

**Once you've created a .m3u8 file/master playlist either yourself or through qencode.com  now you need to store it somewhere online where your app can stream it from.**
I used cloud.google.com (prefer them over aws. Its free the first year and pretty darn cheap afterwards

- I use their "storage" option
- Create a project/bucket
- inside that bucket, make a folder for your newest video. title it something short
- inside that folder upload the new files that have been prepared for streaming

- set your permissions - I just set everything at public but only I could edit.
- give permission for: Storage Object Viewer and to: allUsers
- (https://cloud.google.com/storage/docs/access-control/making-data-public)

Now, once thats all set up, the path to the .m3u8 master file is the link you will enter back in Unity under the Source Path (See section: Setting up the Scene on page 8)

## 1.4 Validating a playlist file to make sure it works

**https://bitmovin.com/docs/player/faqs/how-can-i-validate-my-mpeg-dash-or-hls-stream**

- Get HTTP Live Streaming Tools from apple developer here:
  - https://developer.apple.com/download/more/?=http%20live%20streaming%20tools
- in terminal:
  - mediaStreamValidator [path to where your files are stored]
  - ex:
    - mediaStreamValidator http://storage.googleapis.com/whatever/FHVid.m3u8

# 2.0 EasyAR

**https://www.easyar.com**

- downloads page > EasyAR Sense Unity Plugin (with or without samples)
- once logged in
- Sense Authorization > I need a new Sense License Key
  - Professional or personal
  - app Name: whatever you want
  - Bundle ID: com.[your name].[yourapp] ex: com.BuckyBarnes.ACoolApp
  - make spatial databases, but same name as app Name
- Click on name of whatever you created
  - copy that **license key** or leave page open for when you are ready to enter it in Unity. See step 3.2.1

# 3.0 Unity

**Login to Unity:**
Unity 2020.11f1

- 3D (not universal render pipeline)

**Download**
https://unity3d.com/get-unity/download
**Download Unity Hub**
- Installs tab: Add
  - Unity 2020.1.11f1
  - once finished, back in installs tab, click on the 3 vertical dots next to 2020.11f1
    - add modules
    - click Android or iOS build support, whatever platform you want
      - make sure everything under the expand arrow is selected

## 3.1 Setting up the App for the first time

**(skip to 4.0 if app is already set up)**

## 3.1.1 Unity Build Settings (same settings if building to test, not release)

**iOS:**
Unity hub > installs > click on 3 dots of whatever version of unity you have > add modules > click iOS
(file > build settings)
- Scenes in Build: make sure the current scene is checked, otherwise click add open scenes)
- platform: iOS
- run in xcode: Latest version
- run in Xcode as: Debug
- Symlink Unity libraries: checked (this will not be an option if building on a pc)
- Development build: check
- Compression Method: LZ4HC

**Android:**
file > build settings
- left side: Android
- texture compression: don't override
- Build app bundle (google play): check
- compression method: LZ4HC

## 3.1.2 Project Window + icons + Hierarchy

**Deleting unnecessary assets**
- example assets > delete folder
- Materials > delete Skybox
- tutorial info > delete folder

**Adding assets**
- scenes: rename SampleScene to "Main"
  - Scenes > SampleSceneLightingSettings (Rename to MainSceneLightingSettings)
- make folder named icons > add all icons of logo
  - select all icons you just dragged in
  - inspector window (on right) > advanced > non-power of 2: none (this keeps the specific resolution of the icons, otherwise it would resize them to standard sizes)
  - any transparent icons:
    - inspector > texture type: Sprite
    - inspector > advanced : alpha is transparent (check)

**Hierarchy**
- delete all example prefabs **except: Main Camera**

## 3.1.3 Settings

**edit > project settings > player settings > icon**
- Add company name and project name
- under "cursor hotspot" there is a computer icon and an iOS icon. click iOS
  - icon > add icons if they are ready, if not see section 3.5.2
  - Splash Image > logos > + > select your transparent logo

**edit > project settings > player settings > resolution and Presentation**
- render over native UI: uncheck

**edit > project settings > player settings > Other Settings (iOS)**
- Rendering
  - Dynamic batching: check
  - compute skinning: check
  - auto graphics API: check
- Identification:
  - set signing name/ bundle identifier: com.[your apple developer name].CosmicPerspective, ex: com.BobbyBobbertson.ACoolApp
    - the name here has to match the name you are using to sign the app in Xcode or whatever you're using

        - automatically sign: check
    - Minimum API level: iOS: 11.0
- Configuration
    - Scripting Backend (IL2CPP)
    - Api compatibility level: .Net Standard 2.0
    - camera use description: "This app requires use of the camera. Please allow in phone settings"
    - mute other  audio sources: check
    - unclick optimize mesh data for faster compile times
    - requires persistent WiFi: check
- Optimization
    - Managed Stripping Level: High (can adjust to lower if you don't need to strip for size of the app)
    - Script Call Optimization: Fast but no exceptions (can adjust to slower if wanted)
    - texture mipmap stripping: check

**edit > project settings > player settings > Other Settings (Android)**

- Rendering
    - Auto Graphics API: check
    - multithreaded rendering: uncheck
        - BUT- if you need to use video recording features, set Graphics API to OPEnGL ES 2.0 or 3.0 and cancel multithreaded rendering - android api needs to be lvl 25 or higher
    - static batching: check
    - dynamic batching: check
    - compute skinning: uncheck
- Identification
    - Minimum API level: Kitkat (19)
- Configuration
    - scripting backend: IL2CPP
    - mute other audio sources: check
    - Target architectures, ARMv7 : check
    - Target architectures, ARM64 : check
    - Internet access; require
- Optimization
    - Optimize mesh data: check
    - texture mipmap stripping: check

**Project settings > quality > rendering**

- Realtime reflection Probes: uncheck
- soft particles: uncheck
- shadow: disable shadows
- shadow resolution: low

### 3.1.4 Lighting

**window > rendering > lighting  > scene**
- Realtime lighting: all unchecked
- lightmapping settings: bounces: none
  - russian roulette start bounce: never
  - filtering: auto
- shadow: disable shadows
- shadow resolution: low

**window > rendering > lighting  > environment (next to scene at top)**
- environment lighting: source: color (select black)
- environment reflections
  - source: custom
  - intensity multiplier: 0
- other settings:
  - fog: uncheck
  - halo strength: 0
  - flare speed: 0
  - flare strength: 0

## 3.2 Assets to Install (from Asset Store or External)
### 3.2.1 EasyAR and packages needed to make it build properly

**Download package from https://www.easyar.com/view/download.html**
- drag and drop that download into the unity asset folder/project section
- when prompted, make sure everything is checked and click import

**Unity at the top**
- EasyAR > Change license key: paste license key from in section 2.0

**Edit > Project Settings > XR Plug-in Management**
- enable
- ARKit: Check (if iOS)
- ARCore: Check (if Android)
- **Window > Package Manager**
  - install AR Foundation
  - install ARKit XR Plugin (if iOS)
  - install ARCore XR Plugin (if Android)
  - make sure XR Plugin Management is installed in the package manager
    - if you get an error "CS0117: BuildPipeline does not contain a definition for 'GetBuild TargetName"you'll need to downgrade to a different version that does contain the definition.

- click on arrow next to the package to see more info, then click on versions. Click version 3.2.16, then in the window next to it, Update to 3.2.16 or whatever you think will work

### 3.2.2 AVPro

**window > asset store**
- login > download / import  (from purchases)
- don't need to import demo folder
- in plugin folder, only need either iOS if mac or Android if android

### 3.2.3 DOTween Pro

window > asset store
- login > download (from purchases)
- Open DOTween Utility Panel
- Setup
- Update (apply default settings)
- Create ASMDEF > ok

## 3.3 Manage Assembly Files / ASMDEF

**These are important because Unity runs certain batches of code in an order.** With AVPro, we need to make sure this runs first before any of the other scripts run, so the scripts can reference AVPro's customized namespaces.

### 3.3.1 AVPro Assembly Files

**Scripts folder should look like this:**
- AVProVideo > Scripts: inside will be an Editor folder and a Main folder (you'll have to make the main folder yourself)
  - drag everything in the internal and component folders into the Main folder,
  - Don't touch anything in the editor folder and make sure the editor folder isn't a child of the main folder
  - I usually delete support unless you are using the assets mentioned in Support

- AVProVideo > Scripts > Main > right click > create > Assembly Definition (name it: AVProScripts)

### 3.3.2 EasyAR Assembly Files

**Move all EasyAR scripts within the Resource and Utility folder into parent Scripts folder EXCEPT Editor Folder.**

- EasyAR > scripts > right click create > folder > name it Main
- Main > right click > new assembly definition. name it EasyARScripts
  - Assembly Definition References > use GUIDs: check
  - click the + icon under use GUIDs
    - in that new empty space, drag AVProScripts into it
  - click apply at bottom

## 3.4 Editing Scripts

### 3.4.1 ImageTargetController

**EasyAR > Scripts > ImageTargetController**
(this script is so when the app tracks an object, it can reference the AVPro namespace, and triggers the streamed video to play)

**comment out**

- entire section under OnDestroy()

**add**

in the using header section add:
using RenderHeads.Media.AVProVideo;
under class, add:
public GameObject ResponsiveMediaPlayer;
public GameObject QuadForVideo;
under methods:
start()
add to the end:
QuadforVideo.SetActive(false);
protected override void OnTracking()
after CheckScale(); add:
QuadForVideo.SetActive(true);
ResponsiveMediaPlayer.GetComponent<MediaPlayer>().Play();

**once you update this script and add other components from the next sections into the hierarchy, this script will now have 3 empty fields you can drag game objects to. You can come back to this late but don't forget it!**

- drag the media player into the ResponsiveMediaPlayer field (ex: FHplayer from section 3.6.3)
- drag the quad game object that will be displaying the streamed video (ex: FHVid from section 3.6.6)

## 3.4.2 PauseWhenInactive

**Make new script: PauseWhenInactive**

in the using header section add:

```
using RenderHeads.Media.AVProVideo;
```

under class, add:

```
public GameObject mediaPlayer;
public GameObject Trigger;
```

delete void Start() section

```
void Update()
{
    if (Trigger.activeSelf == false)
    {
        mediaPlayer.GetComponent<MediaPlayer>().Pause();
    }
```

**Attach this script to your Media player object in the hierarchy (ex: FHPlayer from section 3.6.3) by scrolling down in the inspector and hitting "add component. make sure you don't forget to drag the following game objects into the empty fields:**

- drag this same object into its own mediaPlayer field (ex: FHPlayer)
- drag the trigger into the Trigger field (ex: FHTrigger)

## 3.5 Add Materials

**Make a new folder under Assets named Materials. Put all these materials there.**

## 3.5.1 Icons/Logos for game in app stores

**Logos for icons (make a separate folder in the materials section if you want)**
- Import/drag all sizes of logos at different resolutions
- make sure they're named correctly
- **this next part is important because unity compresses images weird!!!** ****
  - advanced: Non-power of 2: None
  - scroll down to bottom: Default: Compression: None
- **edit > project settings > player settings**
  - add icon to "default icon" box (can drag and drop)
  - under "cursor hotspot" there is a section for iOS and Android. Set the icons to whichever app you are building.
    - override for PC, Mac & Linux Standalone: check
    - icon > add icons

## 3.5.2 StreamingAssets folder for triggers

**Make a new folder under Assets named StreamingAssets, exactly as that is written**
- Put all of the images that you want to be tracked / the images that you want to act as triggers into this folder (can drag and drop)
- make sure they are appropriately named like: [name]TriggerImage

## 3.5.3 Transparent Material for video player

**We need a transparent material for the video player mesh**
- Import Transparent PNG file (will be connected to media player under (apply to mesh) this is just literally any size png that is a completely transparent square/rectangle
  - Alpha Is Transparent: check

## 3.6 Hierarchy: SETTING UP THE SCENE! (finally~~)
## 3.6.1 Main Camera

**this is likely already in the scene**
- make sure it is tagged as MainCamera
- Environment
  - Type: Solid Color

- Background: Select Black
- Clipping Planes: 0.1
- Depth: -1

- make sure it has the components attached:
  - Audio Listener

## 3.6.2 EasyAR_ImageTracker-1 prefab

**I rename mine EasyARController because it looks better.**
(get from EasyAR > prefabs > composites: EasyAR_ImageTracker-1)

- Center Mode: First Target
- Center Target: none
- Horizontal Flip Front: None ***
- Assemble Mode: Auto

- > RenderCamera > target camera > drag main camera into here (can leave eternal parameters blank)
- > ImageTracker: Simultaneous Target: however many targets you have

## 3.6.3 AVPro Media Player

**We will need a MediaPlayer for each specific trigger. We wouldn't have one media player streaming all of the videos, and so because we will need many, I name mine specifically so I know which media player belongs to which trigger. EX: if FHTrigger is the trigger, than FHPlayer is the Media player**

- **Add: AVPro > MediaPlayer** (can right click in hierarchy)
  - settings
    - source path: Absolute path or URL
      - add link to wherever you have your .m3u8 files ready for streaming in the cloud (see section 1.3)
    - auto open : yes
    - auto play : no
    - loop : yes
  - **add component: PauseWhenInactive script**
    - drag this same object into its own mediaPlayer field
    - drag the trigger into the Trigger field (ex: FHTrigger from 3.6.4)

## 3.6.4 ImageTarget (the trigger)

**Get from EasyAR > prefabs > Primitives**

- name this whatever corresponds with the image trigger (ex: [name]TriggerImage, like FHTriggerImage)
- Layer: 1: TransparentFX *** Whatever this is, make sure it is NOT set to 0:Default but to something below it. This is because we are putting a sprite (the patch) in front of it, and the sprite will automatically go behind it otherwise. This changes the sorting order to have the patch visible on top of it. Even if it doesn't look like it in the editor, in runtime the patch will come to the front
- set scale properly. usually like, .15 for X Y and Z
- If you edited the script properly in section 3.4.1, you will now have three new blank fields
  - Responsive Media Player: drag the [name]Player Game object we just made in section 3.6.3 into this empty field
  - Quad for video: this is where we will drag a game object that we'll make in section 3.6.5 ([name]Vid)
- Image File Source: Path Type: Streaming Assets
  - if one doesn't exist: go into project folder, right click on assets, create > folder. Name it StreamingAssets
    - This folder is where we drag in our images that we want to track.
  - Image File Source: Path: Type in the name of the image trigger we put in the StreamingAssets folder (ex: FHTriggerImage.jpg)
  - Image File Source: Name: Whatever you want (ex: FHTriggerImage)

## 3.6.5 Streaming Video

**Add as child of Image Target (ex: a child of FHTrigger).**

- Right click on Image Target/Trigger > create > 3D > Quad
  - name: rename [name]QuadVid, ex: FHQuadVid or FHVid
  - Layer: 1: TransparentFX *** Whatever this is, make sure it is NOT set to 0:Default but to something below it. This is because we are putting a sprite (the patch) in front of it, and the sprite will automatically go behind it otherwise. This changes the sorting order to have the patch visible on top of it. Even if it doesn't look like it in the editor, in runtime the patch will come to the front
  - transform: whatever size and space you want your video to be! I would make it match the size of the trigger if you want it to cover the trigger completely.
  - Materials > element 0 = transparentMaterial (from section 3.5)
  - lighting
    - Cast Shadows : off
    - receive shadows : uncheck
  - probes
    - light probes: off

    - reflection probes: off
- **add component: apply to material (should have AVPro logo on it)**
  - drag media player ([name]Player] from section 3.6.3) into media field
- **add component: DOTween Animation**
    - fade
    - duration : 3
    - Delay : 2
    - ease : InOutSine
    - loops : 1
    - from : 0
- **Scroll to bottom of inspector in Material section****
  - shader: AVProVideo / Unlit /Transparent (texture+color+fog+packed....)
  - the two texture boxes should be empty / set to none
- **After this is set up, make sure to drag this game object (this 3D quad in the hierarchy) into these two places:**
  - the [name]Player (from section 3.6.3)
    - under apply to mesh component > Render Target > Mesh: drag into this field
  - the [name]Trigger or target (from section 3.6.4)
    - Quad For Video: drag into this field

### 3.6.6 Make sure everything is referenced properly

**connect player, patch and vid in the empty spaces in the Target/Trigger game object (from section 3.6.4)**

- responsive media player: [name]Player  from section 3.6.3
- QuadForVideo: [name]Vid from section 3.6.5

## 4.0 If you are using the pre-built github template, start here!

**Open the app in unity. Project window > assets > scenes > Main**

### 4.1 Logging in

**Make sure you have an account with Unity and EasyAR**

## 4.2 Streaming videos

**Storage**
I currently have all of this set up and it is set to stream from my own cloud storage bucket. I should have included a zip file of all of those video playlists, so now you'll need to set up your own cloud storage account to host them if you don't have one already.
- See section 1.3 and 1.4 on uploading to cloud storage and validating the links to make sure they work!

**Updating the video path urls to their new storage urls**
- Open up Unity and in the hierarchy window, you'll see lots of game objects called [name]Player
  - (([name] is just the catchall for different names))
  - click on each of the different players, and under Media Player > Source Path
    - (Right under it, it should say Absolute Path or URL). In the text field below that, please update each streaming video playlist url to the new one.

## 4.3 Adding up new triggers to the hierarchy after everything is already set up

**Update the number of tracked targets in the Image tracker**
- EasyARController >ImageTracker game object
  - (ImageTracker is the child game object of the EasyARController parent game object)
  - ImageTracker: Simultaneous Targets: whatever the new number is.

**Preparing and storing the new video to be streamed with the new trigger**
- see section 1.0

**Uploading new trigger:**
- Project window: Assets > StreamingAssets folder
  - Put all of the images that you want to be tracked / the images that you want to act as triggers into this folder (can drag and drop here)
  - make sure they are appropriately named, like [name]TriggerImage
  - see

**Copying the [name]Trigger and [name] from the hierarchy**
- select, copy and paste two [name]Trigger and [name]Player games objects that have the same [name] and paste it back into the hierarchy
- rename both parents and children: change the names to [new name]Trigger and [new name]Player
  - in children: change [name]Vid to [new name]Vid

**[new name]Trigger**
- Make sure Responsive Media Player, and Quad for video fields are populated with the new name equivalents
- Image File source: Path: update paths to the new trigger image found in the StreamingAssets folder
- Name: change the name to the new name
- See section 3.6.4 if you need more details about the [name]Trigger game object

**[new name]Vid**
- Transform: adjust the height/width/scale as needed to fit the size of the new trigger image. This vid game object will play the streamed video, and it needs to be slightly bigger than the trigger image. Once you've got it, hit the play button and adjust as needed. You might have to adjust again once you test out the app on a mobile device
- see section 3.6.5 if you need more details about the [name]Vid game object
- if you see an empty rectangle that jumps around, thats okay! it shouldn't show up in deployment. I bought the iOS and Android license for AVProVideo and the sdk is recognizing that it isn't being run on iOS or Android but instead on a PC and thus is trying to throw up a watermark. Once its deployed it'll be gone.

**[new name]Player**
- **IF USING TEMPLATE AND BUILDING FOR ANDROID**, delete and install AVPro Video (android) from the asset store. The one included in this template is for iOS.
- Source Path: please update to the exact external URL of your streaming video
- Auto Open: checked
- Auto Play: unchecked

**Test it out in the editor!**

# 5.0 Testing on a device simulator

## 5.1 In Unity (Recommended)

**Device Simulator**
- Window > Package Manager
  - click gear icon > advanced project settings
    - Advanced settings: enable preview packages: check
- Window > package manager
  - Scroll until you see Device Simulator (you should see a yellow box next to it that says "Preview"
    - click install
- Window > general > device simulator

- now you can click between the simulator and the game window when you press "play.
  - after hitting play, just click on the simulator
- top left corner of simulator: select the type of device you want to simulate

## 5.2 Android Studio

**(I used to use this to test/simulate on different types of virtual android devices before the unity simulator)**

- Drag and drop your .apk file (you will have to build it first, see section 6.0) from unity into android studio
  - once it loads in and you see the green hammer icon up top light, click on it to build
- tools > ADV Manager
  - +create Virtual Device (add whatever devices you want to test on. I usually do the earliest (in this case, Kitkat API 19), the latest, and one in the middle)
  - press the green triangle icon next to the device you want to test on
    - when it loads that emulator (give it a minute) flip it to the position you want it to be on using the white and grey buttons to the side)

# 6.0 Building / exporting the app for test/runtime

## 6.1 Buttoning up project/reducing file size:

**Doing this reduces bloat and the end file size**

- edit > project settings > graphics > Shader Stripping
  - light map modes > custom
    - untick all boxes.
  - fog modes > custom
    - untick all boxes; instancing variants > strip unused
- edit > project settings > player > other > optimization:
  - strip engine code: check
  - managed stripping level: high
  - script call optimization: fast but no exceptions (you can change this and the managed stripping level if either makes your project not work, lol)
- build settings: if you are building on a mac (which... how? EasyAR doesn't work with mac yet) select symlink unity libraries. lol only including this so to remember for other things
- project window: can delete any "demo" folders in AVPro / Demigant / EasyAR folders

- Window > package manager: you can delete unnecessary packages that you aren't using like unity collaborate or timeline

## 6.2 Build and Run/Test

### 6.2.1 iOS

**file > build settings > save it as whatever you want**

- Scenes in Build: make sure the current scene is checked, otherwise click add open scenes)
- platform: iOS
- run in xcode: Latest version
- run in Xcode as: Debug (if debug),  Release (if release)
- Symlink Unity libraries: check (only if you're building on a mac, which you aren't because EasyAR is not compatible with mac yet, so ignore this)
- Development build: check (if debug), unchecked (if release)
- Compression Method: Lz4HC

### 6.2.2 Android

**file > build settings > save it as whatever you want**

- Scenes in Build: make sure the current scene is checked, otherwise click add open scenes)
- platform: Android
- texture compression: don't override
- ETC2 fallback: 32-bit
- Development build: checked if debug, unchecked if release
- Compression Method: Lz4HC

# 7.0 Preparing for Deployment / testing on a mobile device

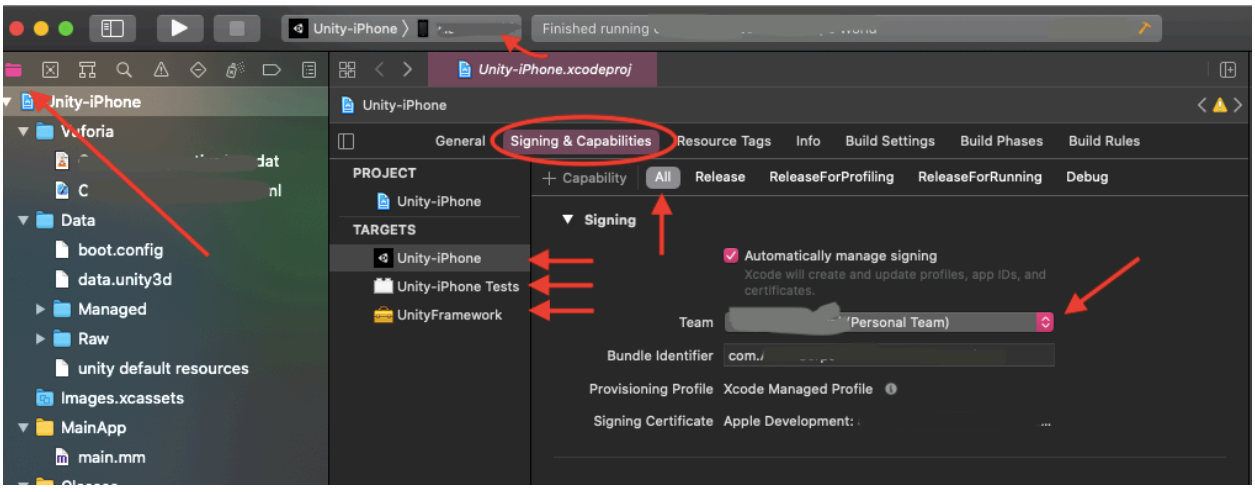
**You'll need to change the bundle identifier to your developer name in editor > project settings > player > other > identification: com.[your developer name].NameOfYourApp**

## 7.1 iOS
### 7.1.1 Xcode - testing on a device

**(see image on next page)**
Copy that folder that you just built and send it however you want to a mac computer to use Xcode. If you don't have a mac, there are a few solutions for you to get Xcode on a pc. see codewithchris.com/xcode-for-windows/

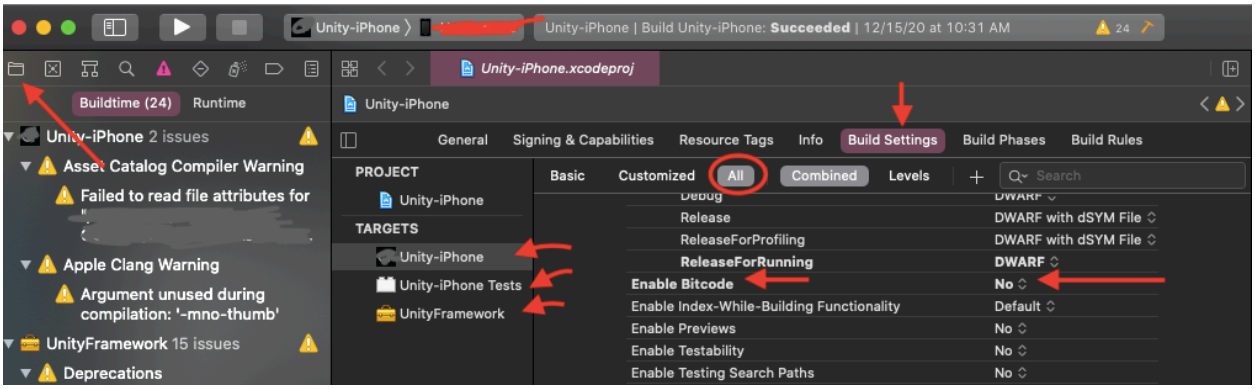
open it and click on Unity-iPhone-Xcodeproj

- it should open in Xcode
- Click on folder icon in top left corner
- Click on folder icon in top left corner
  - in the second window pane, you'll see PROJECT with Unity-iPhone under it, and TARGETS, with a few targets under it.
- click each of the targets and select"signing & capabilities" to make sure you are signed in on all three.
  - select your "team" + check Automatically manage signing
    - if you don't have a team set up yet:
      - Xcode > preferences > accounts > + button on bottom left to add your apple id > select Apple ID > follow instructions
  - if testing: click on "debug" (note: there are some weird bugs with the latest few Xcode's and using it to run an app in debug mode unless you have a developer account, its REAL frustrating).
    - connect your phone via usb to laptop
    - make sure your phone is selected in the place to test the app (red arrow in pic, my phone is called Hope World)
  - if releasing, use your pro developer account (can get at developer.apple.com)



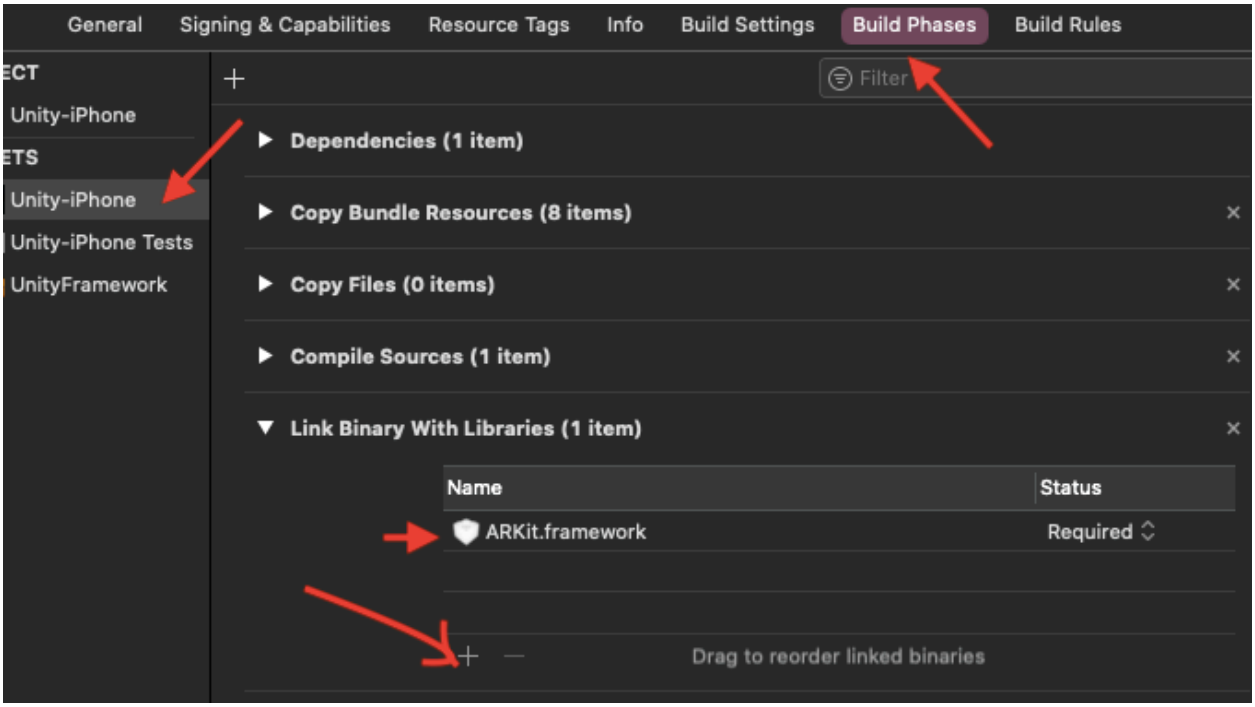

**Click folder icon > Build settings > build options**

- under each of the targets, click build settings
- click on each of the targets
  - scroll down until you see the "build options category
  - enable bitcode: select "no"




**Click folder icon > Build Phases**

- click the the first target
  - Link binary with libraries
  - hit + button
  - select ARKit.framework




**Building**

- Press the "play" icon (sideways right triangle at top left)
- usually on your device you'll have to enable permissions to use the app:
  - on iPhone: settings> general > scroll to bottom, Device Management > click trust developer or whatever

If you're getting errors on your build, see section 8.5

### 7.1.2 Submitting your app to the app store

**This describes the process much better than I could**
https://codewithchris.com/submit-your-app-to-the-app-store/

**another tutorial that can explain it better than me**
https://learn.unity.com/tutorial/publishing-for-ios#

## 7.2 Android

### 7.2.1 Testing app on a device

**This tutorial explains it better than I could:**
https://learn.unity.com/tutorial/building-for-mobile#5c7f8528edbc2a002053b4a2

### 7.2.2 Deploying to the google play store

**https://play.google.com/console**
- Make an account
- pay the $25 dollar fee to publish an unlimited number of android games to Google Play
- Upload your game to the console and fill out the details, then hit publish!

# 8.0 FAQs

## 8.1 The animations aren't working! (fading in and out/scaling)

If animations aren't working properly its because the material settings and shaders are improperly configured. check out section 3.5.1

## 8.2 The logo icons are compressing into weird sizes!

If logo icons are coming out weird sizes its because ...
check out section 3.5.2

## 8.3 Whenever I hit play in editor, everything is flipped horizontally!

if your in-game preview looks normal but when you hit the play button and every thing is flipped horizontally, check your EasyARController/EasyAR_ImageTracker-1 prefab and see if Horizontal Flip Front is set to World or None. For some reason the default is World and it will flip everything!! Change it to none. Why would they do that.

## 8.4 Why am I seeing an empty rectangle move around where the video is playing?

If you see an empty rectangle that jumps around, thats okay! it shouldn't show up in deployment. I bought the iOS and Android license for AVProVideo and the SDK is recognizing that it isn't being run on iOS or Android but instead on a PC and thus is trying to throw up a watermark. Once it's deployed it'll be gone.

## 8.5 XCODE ERRORS

### 8.5.1 Code signing is required for product type 'Application' in SDK 'iOS whatever'

https://stackoverflow.com/questions/37806538/code-signing-is-required-for-product-type-application-in-sdk-ios-10-0-stic
general: each of the targets: signing & capabilities : "all" NOT release, or REleaseFor Profiling, etc, but ALL**** its important that that one is selected. go through each of the targets and make sure they are all signed. if there is an error, it will be next to the one that isn't signed.

### 8.5.2 If you get a "permission denied" error

there are 3 different ways you can try and solve.
look at the error and copy the file name or path that leads to the file folder
open up terminal terminal
**((****DONT DO ANY OF THESE WITHOUT THE FILE NAME AFTER IT!!!! you are changing permissions so DONT just be like "chmod 777" and hit enter - baaaaad! you don't wanna give permission to everything lol)))**

**try one of these three things:**

- chmod +x (paste file name here, without parentheses)
- chmod -R 755 (paste path up to problem folder name here, without parentheses))
- chmod 777 (paste file name here without parentheses)

## 8.5.3 not found for architecture arm64 easyar-arm64-master.o  or Symbol(s) not found for arm64

- Make sure you've added the "ARKit.framework" in the build phases section of Xcode. see section 7.1.1
- Make sure you've enabled the ARKit XR plugin, AR Foundation Package, and XR Plugin Management in the package manager of Unity
  - honestly I don't know why these work, but my Xcode wouldn't compile unless I had these in there, which :) was :) really :) fun :) to :) figure :) out :)
  - WITH THE XR PLUGIN MANAGEMENT PACKAGE: you might need to downgrade if you get an error "CS0117: BuildPipeline does not contain a definition for 'GetBuild TargetName"
    - You need to downgrade to an XR plugin Management version that does include it. On the package in the Package Manager, click on down arrow to see more info, then click on versions. Click version 3.2.16, then in the window next to it, then"Update to 3.2.16"
- Make sure you've enabled the XR Plug-in Management section of unity > project settings > XR Plug-in Management and checked ARKit