Images2Video Document

Current Version	4.0.2
Current Update	2018/1/20
Author	Isaac Cheng

1. Description :

Images2Video is an assistant plugin to convert serial images to mp4 format video.
We wrapper the all native render APIs in `VideoConverter.cs` and create three example demos to help you to integrate Images2Video plugin into your Unity3D project.

---

2. Required :

- For iOS, requires **SDK 8.1 or higher** to support using **Metal/OpenGL ES 3.0** to fetch the render texture data.
- For Android, requires **API 19 or higher** to support using **OpenGL ES 2.0/3.0** to fetch the render texture data.

---

3. Important :

Please make sure the following settings are correct,

1. `Player Settings>Other Settings>Graphic APIs` is set as OpenGL ES 2 or 3
    a. **Not all Android devices above 4.3 support OpenGL ES 3.0, it also depends on GPU!!**
2. `Player Settings>Other Settings>Write Permission` is set as External(SD Card)

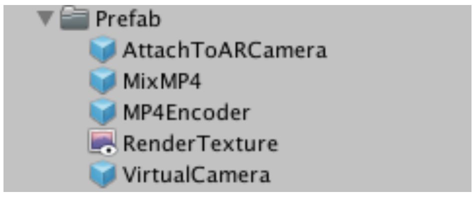
---

4. Demos :

Three simple scenes are included with sample scripts demonstrating its functionality.

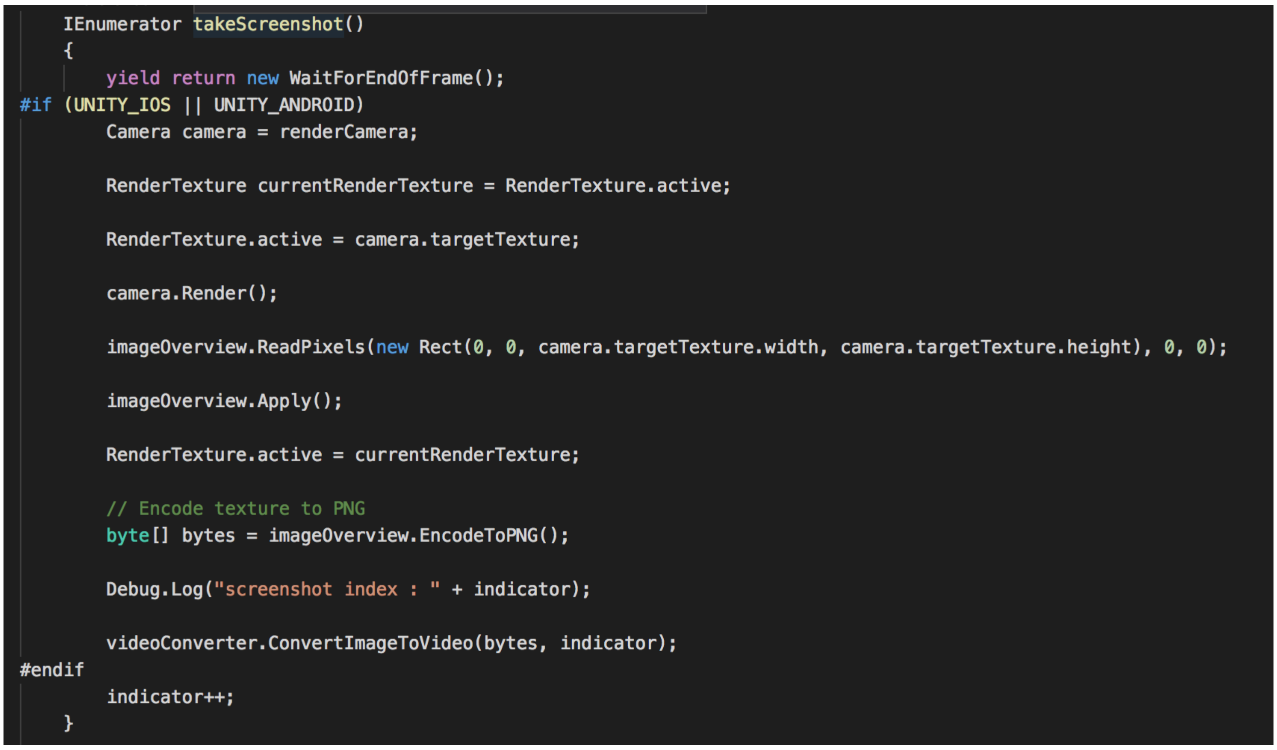
1. ScreenshotExample
2. MP4EncoderExample
3. VirtualCamera
4. VuforiaExample

We wrapper them into prefabs which you can drag&drop in your scene.



4-1. ScreenshotExample :

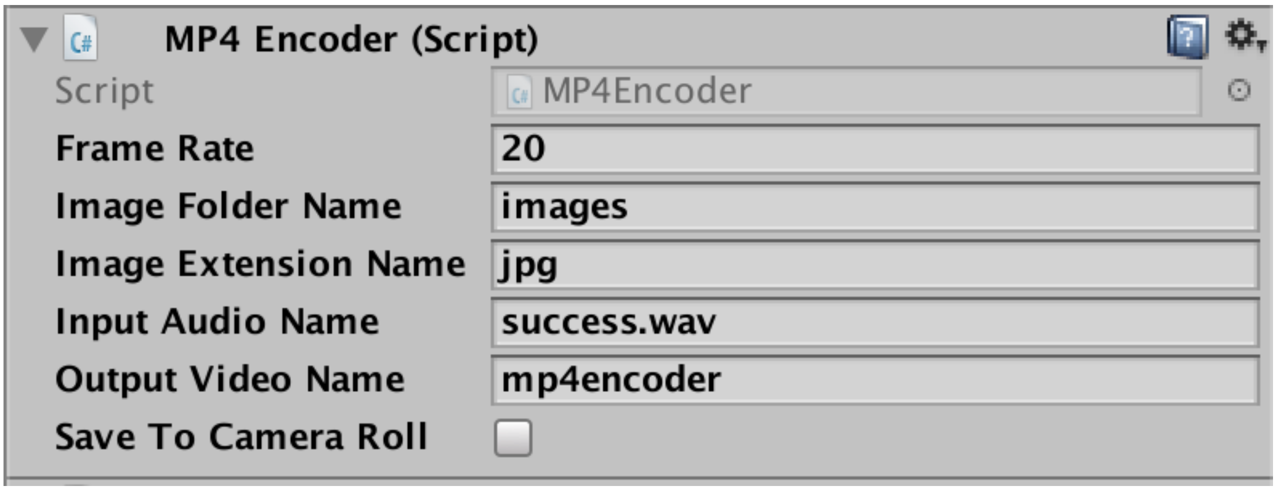
This example implements the traditional process which is posted on Unity3D forum. The performance for this process is unacceptable while calling `Texture2D.ReadPixels()` continuing. If you just want to take one or two screenshots you can use this example.



```
    IEnumerator takeScreenshot()
    {
        yield return new WaitForEndOfFrame();
#if (UNITY_IOS || UNITY_ANDROID)
        Camera camera = renderCamera;

        RenderTexture currentRenderTexture = RenderTexture.active;

        RenderTexture.active = camera.targetTexture;

        camera.Render();

        imageOverview.ReadPixels(new Rect(0, 0, camera.targetTexture.width, camera.targetTexture.height), 0, 0);

        imageOverview.Apply();

        RenderTexture.active = currentRenderTexture;

        // Encode texture to PNG
        byte[] bytes = imageOverview.EncodeToPNG();

        Debug.Log("screenshot index : " + indicator);

        videoConverter.ConvertImageToVideo(bytes, indicator);
#endif
        indicator++;
    }
```

4-2. MP4Encoder :

This example implements to convert serial images to video and merge with the audio.



MP4 Encoder (Script)

Script	MP4Encoder
Frame Rate	20
Image Folder Name	images
Image Extension Name	jpg
Input Audio Name	success.wav
Output Video Name	mp4encoder
Save To Camera Roll	☐

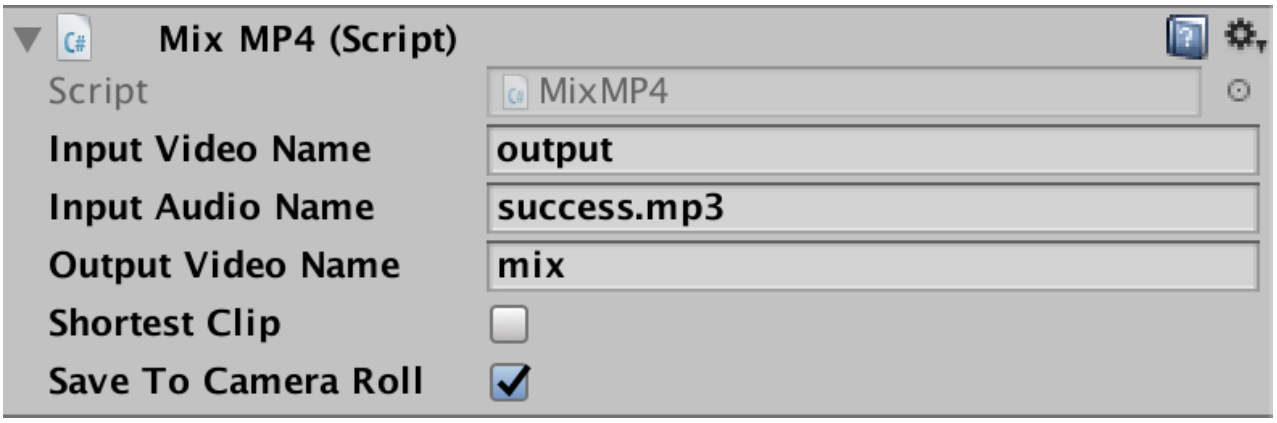
MP4Encoder properties

- **Frame Rate** : The video frame rate.
- **Image Folder Name** : Image folder which all images are put in. The parent folder is `StreamingAssets`.
- **Image Extension Name** : The image extension name. You can set as jpg or png.
- **Input Audio Name** : The audio file you want to merge into the video. The parent folder is `StreamingAssets`.
- **Output Video Name** : The converted video file name.
    - On Android,
        1. Customize the output path: /<folder>/filename.
        2. If you do not customize the path, the converted video file will be saved at `Movies` folder.
    - On iOS, the converted video file will be saved at `<Application>/Library/Caches` folder.
- **Save To Camera Roll** : If checked, the converted video file will be saved into device gallery.

You can also change the parent folder of the image folder and audio file to other place. Please check `MP4Encoder.cs`.

4-3. MIXMP4 :

This example implements merge the video with the audio.



Mix MP4 (Script)

Script	MixMP4
Input Video Name	output
Input Audio Name	success.mp3
Output Video Name	mix
Shortest Clip	☐
Save To Camera Roll	✓

MIXMP4 properties

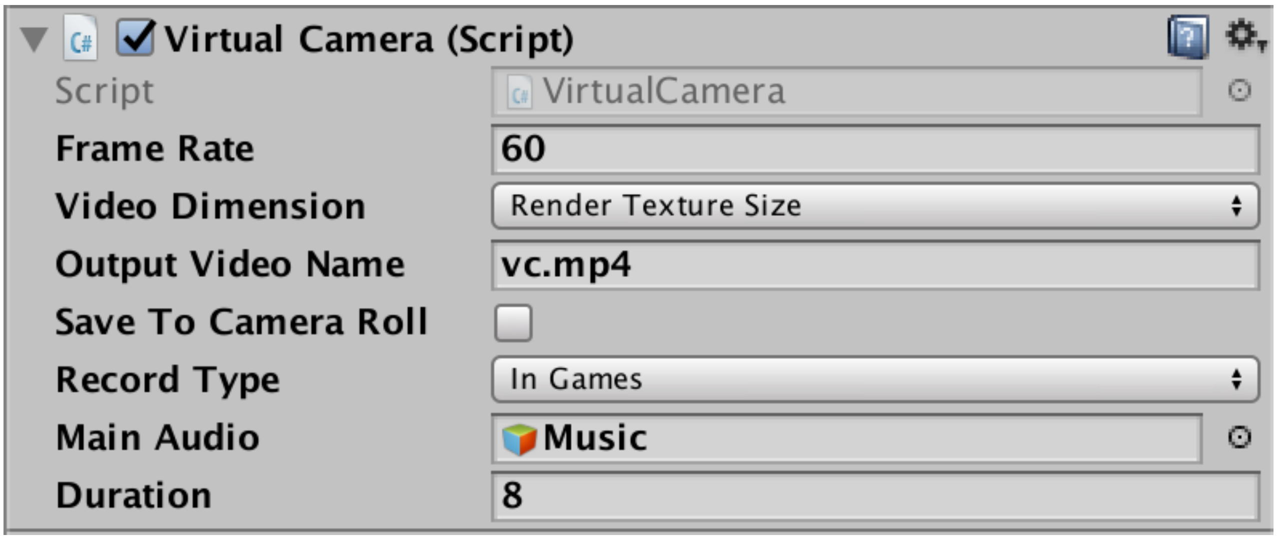
- **Input Video Name** : The video you want to merge. The parent folder is `StreamingAssets`.

- **Input Audio Name** : The audio you want to merge. The parent folder is `StreamingAssets`.
- **Output Video Name** : The converted video file name.
  - On Android,
    1. Customize the output path: /<folder>/filename.
    2. If you do not customize the path, the converted video file will be saved at `Movies` folder.
  - On iOS, the converted video file will be saved at `<Application>/Library/Caches` folder.
- **Save To Camera Roll** : If checked, the converted video file will be saved into device gallery.

You can also change the parent folder of the video and audio file to other place. Please check `MixMP4.cs`.

4-4. VirtualCamera :

This example implements native render APIs to render texture with high performance.

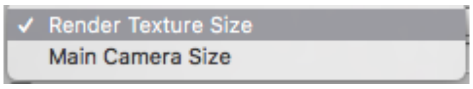


Virtual Camera (Script)

Script	VirtualCamera
Frame Rate	60
Video Dimension	Render Texture Size
Output Video Name	vc.mp4
Save To Camera Roll	☐
Record Type	In Games
Main Audio	Music
Duration	8

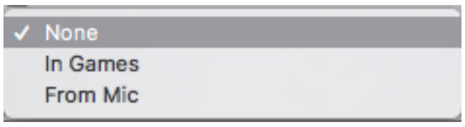
VirtualCamera properties

- **Frame Rate** : The video frame rate. The default value is 30.
- **Video Dimension** : The converted video dimension(width x height). The default is Render Texture Size. How to set the dimension of the render texture, please check section 4-4-1



Video Dimension

- **Output Video Name** : The converted video file name.
  - On Android,
    1. Customize the output path: /<folder>/filename.
    2. If you do not customize the path, the converted video file will be saved at `Movies` folder.
  - On iOS, the converted video file will be saved at `<Application>/Library/Caches` folder.
- **Save To Camera Roll** : If checked, the converted video file will be saved into device gallery.
- **Record Type** : Record the audio then merge into the video. The default is None.



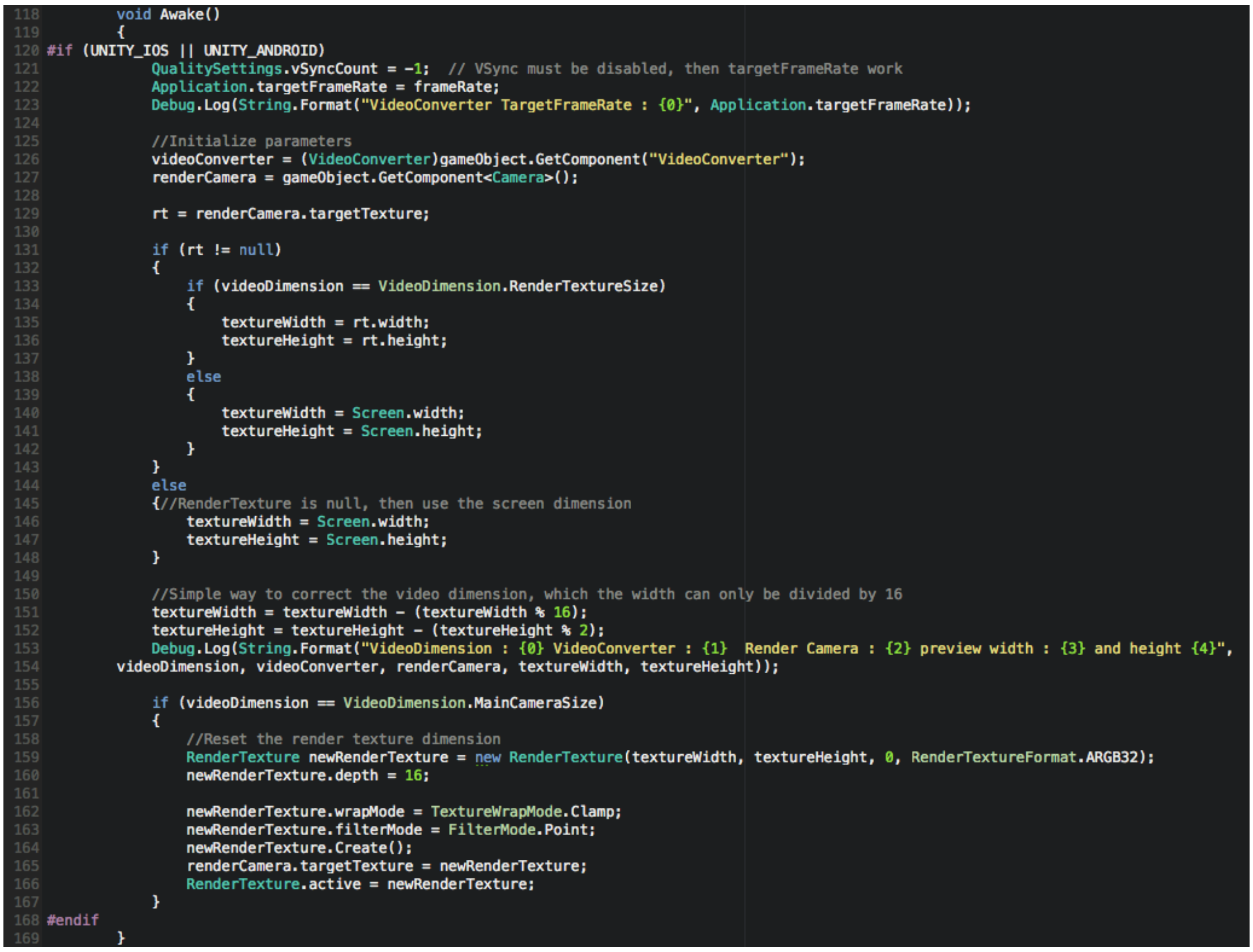
Record Type

- **In Games** : The converted video will merge with game music.
- **From Mic** : The converted video will merge with microphone audio.
- **None** : The converted video will not merge with audio.

- **Main Audio** : Attach to the main game music to control the volume of the game music while recording from microphone. The plugin will downgrade the volume to prevent disturbing microphone recording.
- **Duration** : The length of the converted video. The default is zero which means this property does not work. If you set the duration, then the record process will generate required frames and convert the video with this duration.

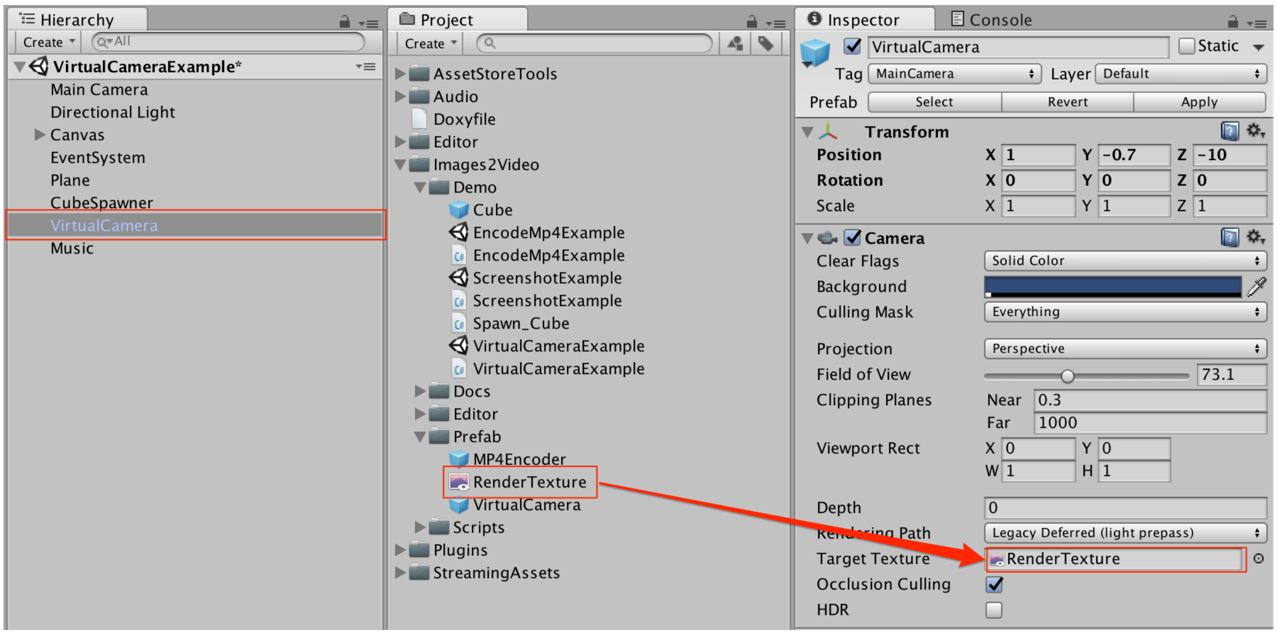
4-4-1 Set the render texture :

Beside modify the dimension of the render texture programmatically,



```
        void Awake()
        {
#if (UNITY_IOS || UNITY_ANDROID)
            QualitySettings.vSyncCount = -1;  // VSync must be disabled, then targetFrameRate work
            Application.targetFrameRate = frameRate;
            Debug.Log(String.Format("VideoConverter TargetFrameRate : {0}", Application.targetFrameRate));

            //Initialize parameters
            videoConverter = (VideoConverter)gameObject.GetComponent("VideoConverter");
            renderCamera = gameObject.GetComponent<Camera>();

            rt = renderCamera.targetTexture;

            if (rt != null)
            {
                if (videoDimension == VideoDimension.RenderTextureSize)
                {
                    textureWidth = rt.width;
                    textureHeight = rt.height;
                }
                else
                {
                    textureWidth = Screen.width;
                    textureHeight = Screen.height;
                }
            }
            else
            {//RenderTexture is null, then use the screen dimension
                textureWidth = Screen.width;
                textureHeight = Screen.height;
            }

            //Simple way to correct the video dimension, which the width can only be divided by 16
            textureWidth = textureWidth - (textureWidth % 16);
            textureHeight = textureHeight - (textureHeight % 2);
            Debug.Log(String.Format("VideoDimension : {0} VideoConverter : {1}  Render Camera : {2} preview width : {3} and height {4}",
        videoDimension, videoConverter, renderCamera, textureWidth, textureHeight));

            if (videoDimension == VideoDimension.MainCameraSize)
            {
                //Reset the render texture dimension
                RenderTexture newRenderTexture = new RenderTexture(textureWidth, textureHeight, 0, RenderTextureFormat.ARGB32);
                newRenderTexture.depth = 16;

                newRenderTexture.wrapMode = TextureWrapMode.Clamp;
                newRenderTexture.filterMode = FilterMode.Point;
                newRenderTexture.Create();
                renderCamera.targetTexture = newRenderTexture;
                RenderTexture.active = newRenderTexture;
            }
#endif
        }
```

you can also change it in Unity Editor.



VirtualCamera Static
Tag MainCamera Layer Default
Prefab

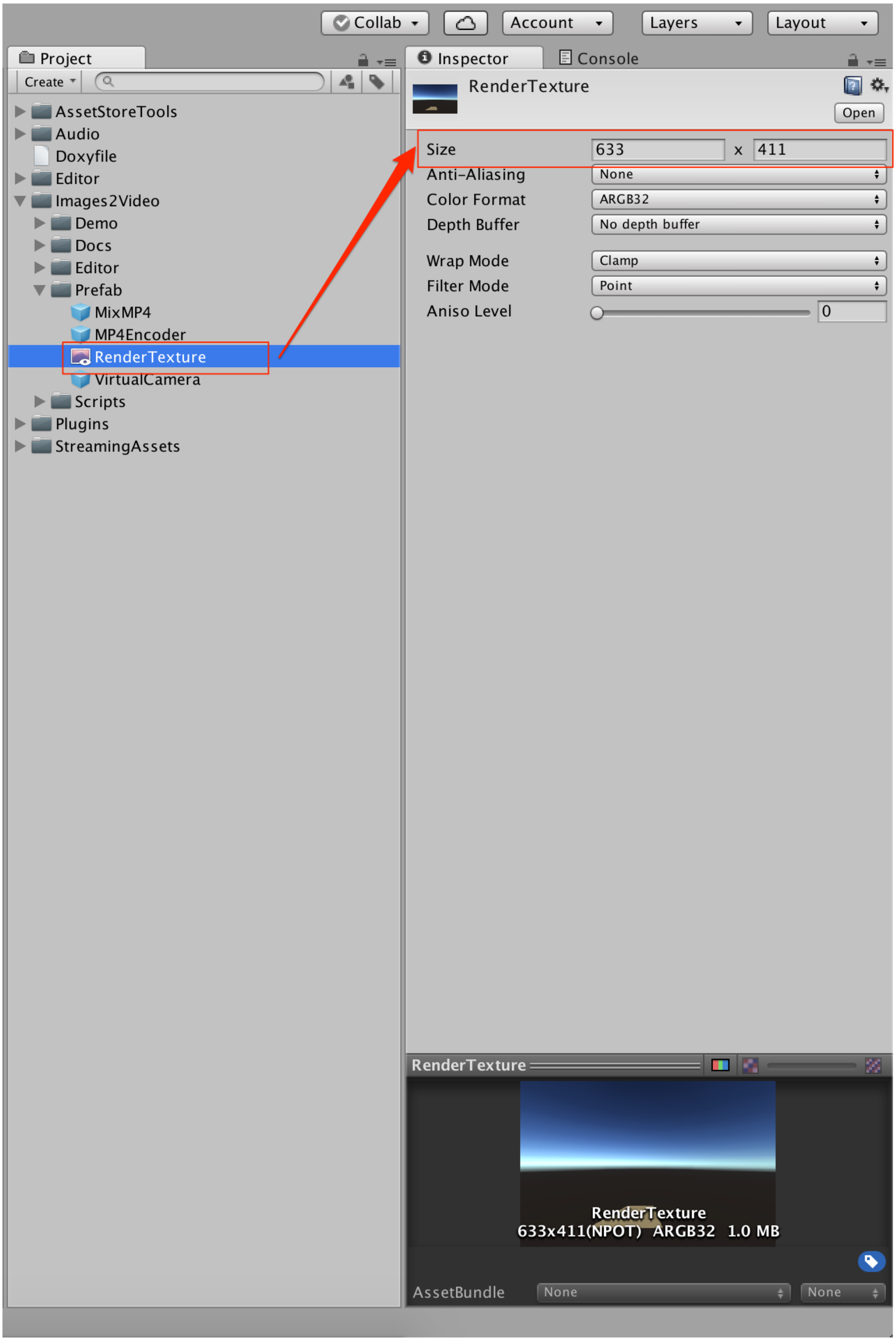
Transform

Position	X	1	Y	-0.7	Z	-10
Rotation	X	0	Y	0	Z	0
Scale	X	1	Y	1	Z	1

Camera

Clear Flags	Solid Color
Background	
Culling Mask	Everything
Projection	Perspective
Field of View	73.1
Clipping Planes	Near 0.3
	Far 1000
Viewport Rect	X 0 Y 0
	W 1 H 1
Depth	0
Rendering Path	Legacy Deferred (light prepass)
Target Texture	RenderTexture
Occlusion Culling	✓
HDR	☐

RenderTexture



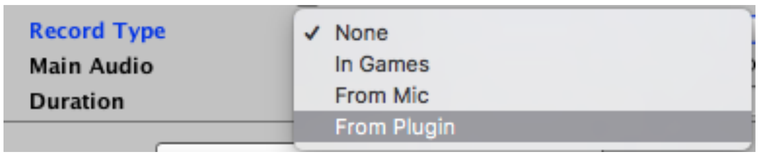
Size 633 x 411
Anti-Aliasing None
Color Format ARGB32
Depth Buffer No depth buffer
Wrap Mode Clamp
Filter Mode Point
Aniso Level 0

RenderTexture
633x411(NPOT) ARGB32 1.0 MB

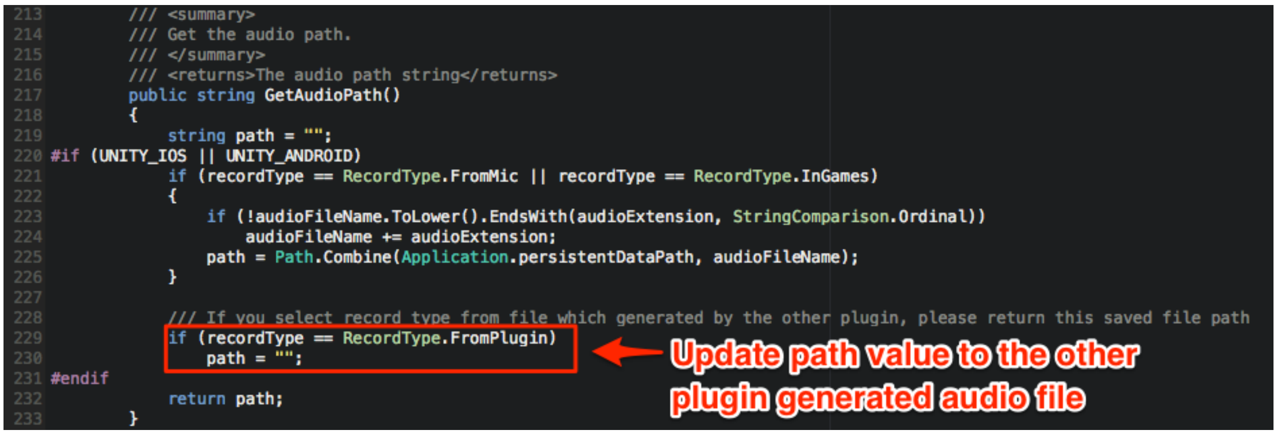
AssetBundle None None

4.4.2 Merge the other plugin generated audio file

If you select record type `FromPlugin`, the converter can merge the other plugin generated audio file.



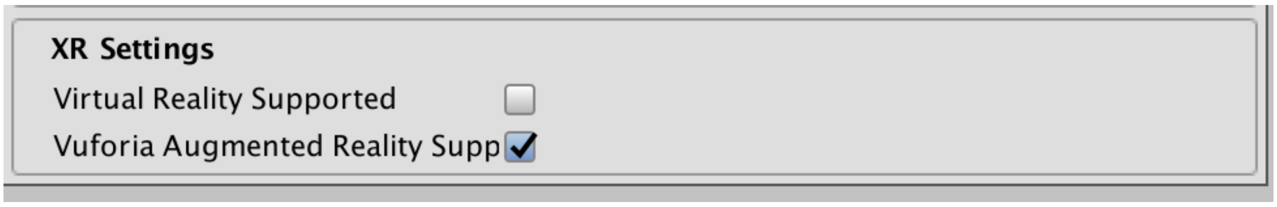
You also need to modify `GetAudioPath` which returns the audio file path in `VirtualCamera.cs`



```
        /// <summary>
        /// Get the audio path.
        /// </summary>
        /// <returns>The audio path string</returns>
        public string GetAudioPath()
        {
            string path = "";
#if (UNITY_IOS || UNITY_ANDROID)
            if (recordType == RecordType.FromMic || recordType == RecordType.InGames)
            {
                if (!audioFileName.ToLower().EndsWith(audioExtension, StringComparison.Ordinal))
                    audioFileName += audioExtension;
                path = Path.Combine(Application.persistentDataPath, audioFileName);
            }

            /// If you select record type from file which generated by the other plugin, please return this saved file path
            if (recordType == RecordType.FromPlugin)
                path = "";
#endif
            return path;
        }
```

4.5 Vuforia Example

First enable `Vuforia Augmented Reality Support` in `PlayerSettings>XR Setting`

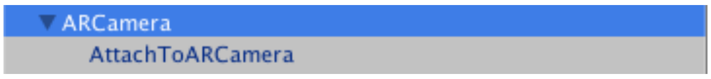


XR Settings

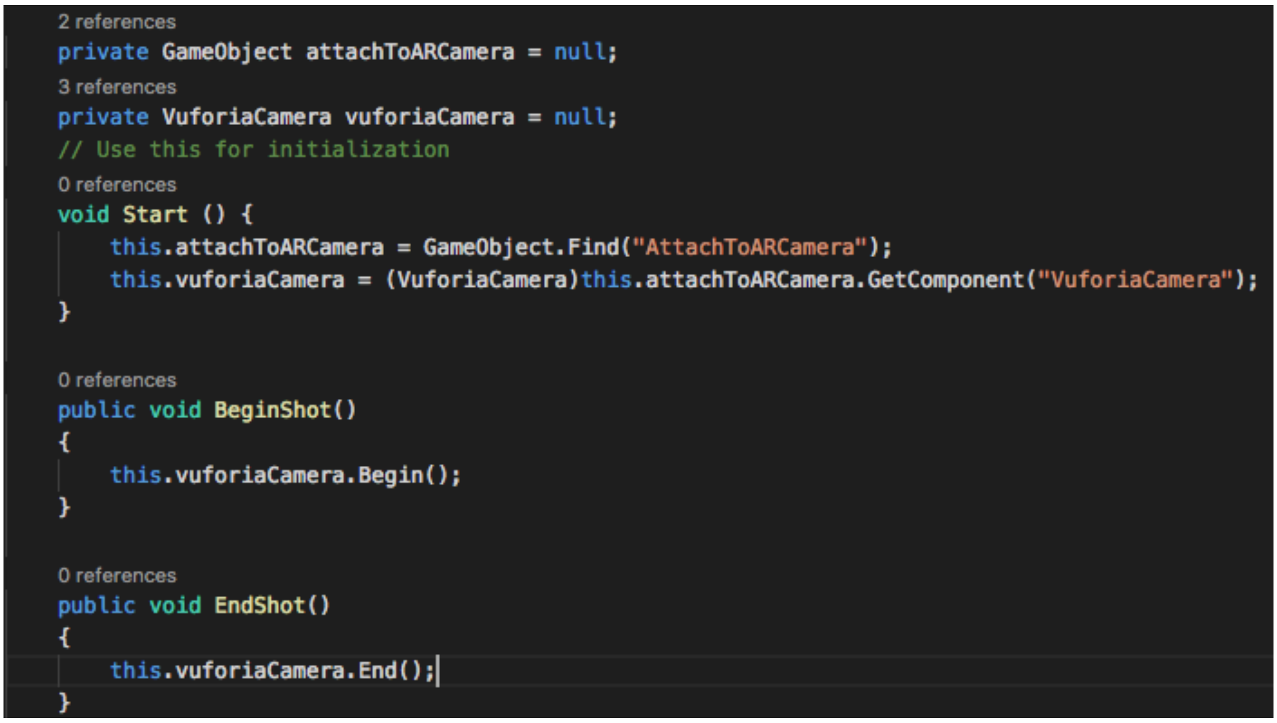
Virtual Reality Supported ☐

Vuforia Augmented Reality Supp ✓

Second drag and drop `AttachToARCamera` prefab into `ARCamera` object



Finally check `VuforiaExample.cs` and see how to start to capture the screen and stop it.



```
2 references
private GameObject attachToARCamera = null;
3 references
private VuforiaCamera vuforiaCamera = null;
// Use this for initialization
0 references
void Start () {
    this.attachToARCamera = GameObject.Find("AttachToARCamera");
    this.vuforiaCamera = (VuforiaCamera)this.attachToARCamera.GetComponent("VuforiaCamera");
}

0 references
public void BeginShot()
{
    this.vuforiaCamera.Begin();
}

0 references
public void EndShot()
{
    this.vuforiaCamera.End();
}
```

Then the video will be generated automatically.

---

5. Note:

If you need help please do not hesitate and send the mail to service@championtek.com.tw.