EasyMovieTexture Manual For Android

Format & Resolution

Supported format :

Android : https://developer.android.com/guide/appendix/media-formats.html?hl=ko

iOS :
https://developer.apple.com/library/content/documentation/Miscellaneous/Conceptual/iPhoneOSTechOverview/MediaLayer/MediaLayer.html

Supported resolutions:

Android: General devices support up to 1920 * 1080.

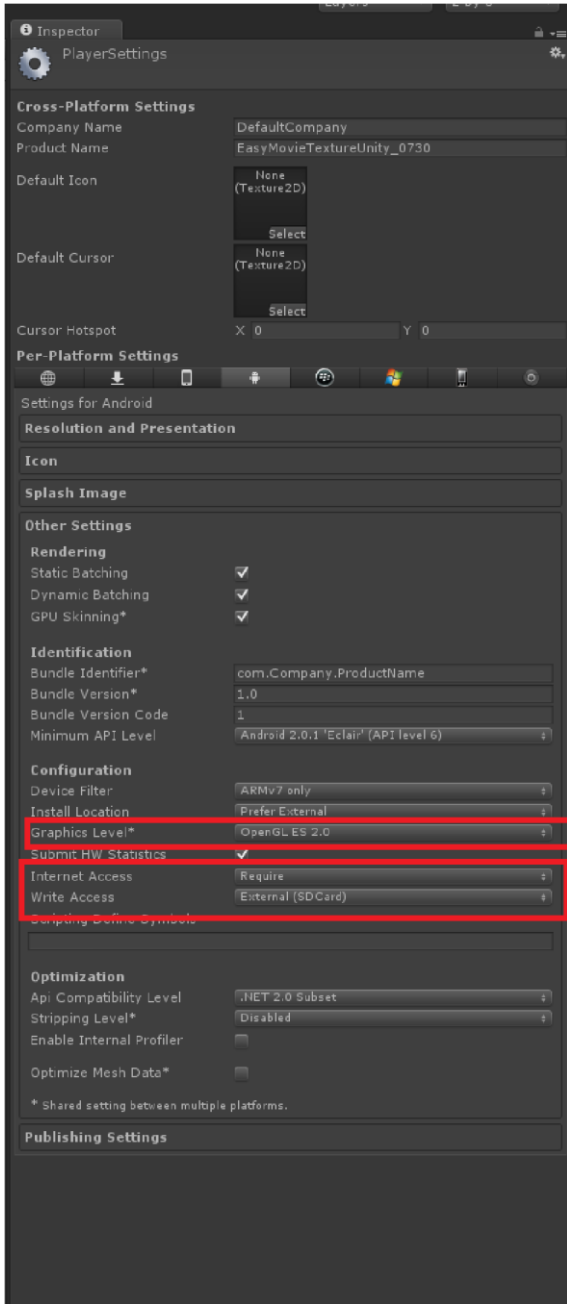The latest device supports up to 4k.

iOS: General devices support up to 1920 * 1080.

The latest device is support up to 2560 * 1440.

iPhone 6s Plus supports up to 4k.

Project Settings

First, you need to configure the player as follows to play the streaming videos and videos stored in the SD card.
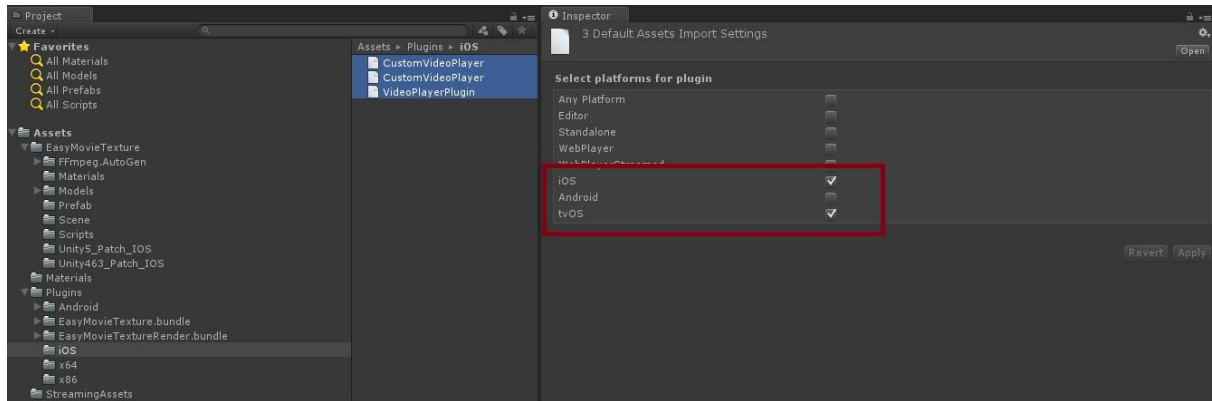(File -> BuildSettings -> PlayerSettings -> OtherSettings)

2. In the case of iOS, Unity 4.X, another of the patch is required. EasyMovieTexture folder Unity463_Patch_IOS. Please import a patch.

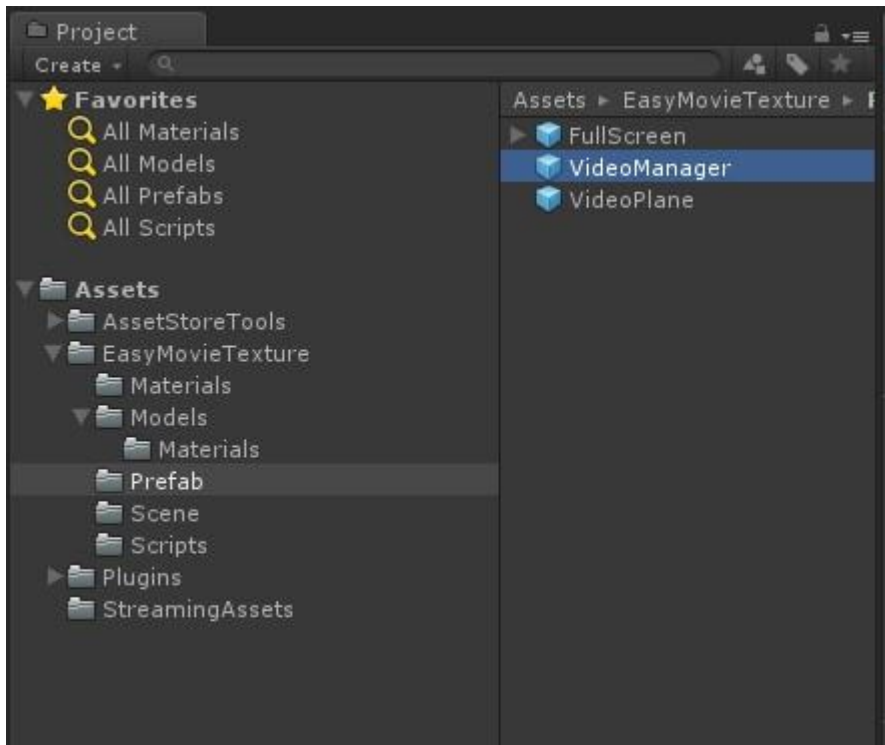## 3. When using the debug mode, set as follows.
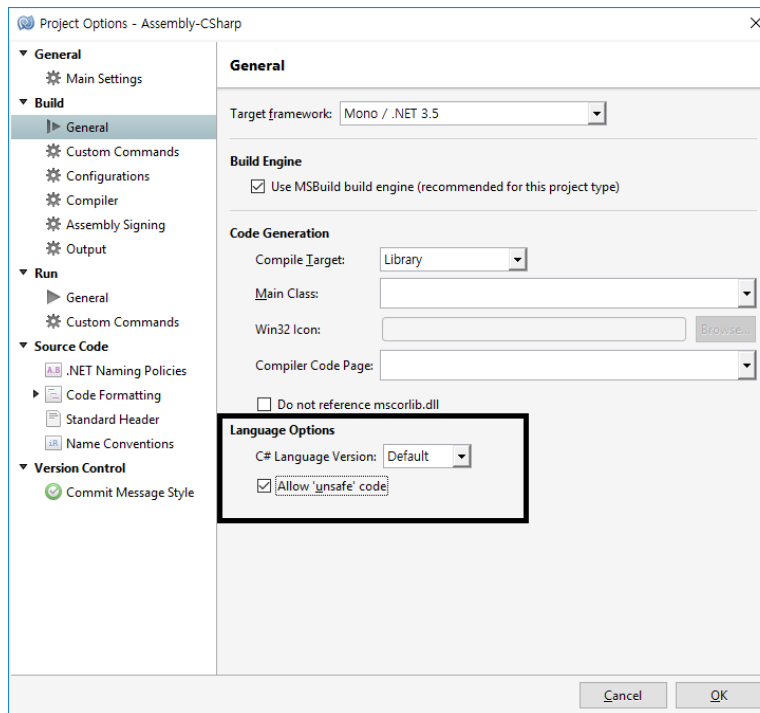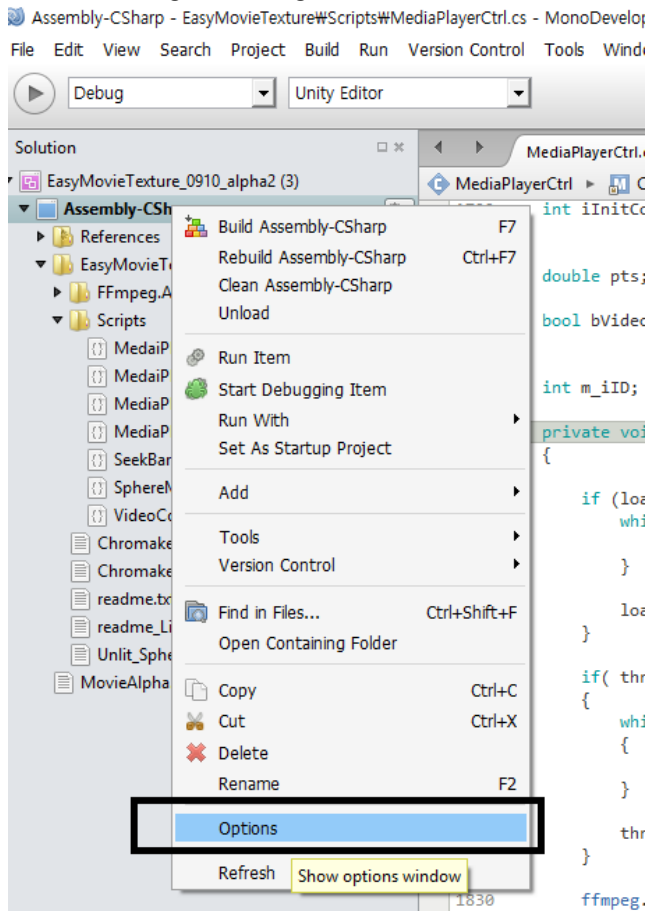
How to set up Apple TV

1. AppleTV(tvOS)support from more than unity 5.3.

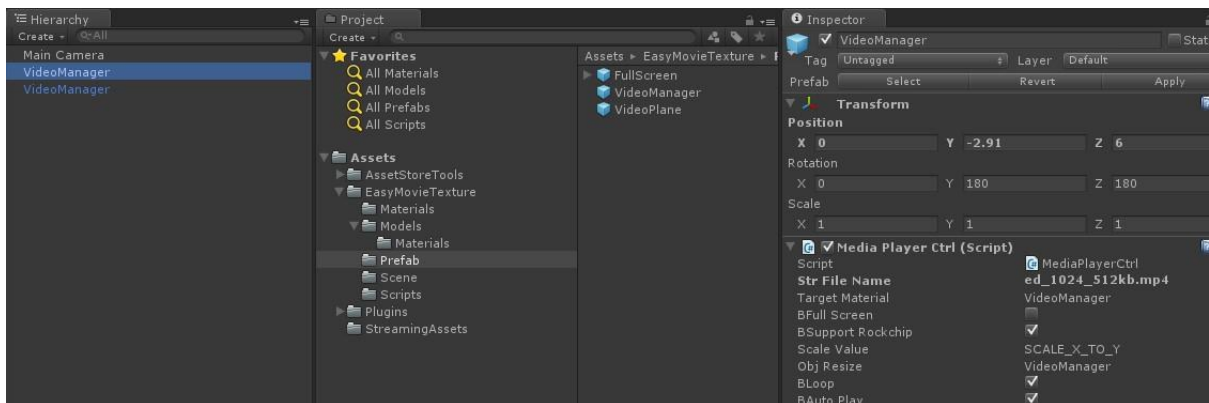2. Please set as follows.

Using VideoManager

First, select VideoManager in the Prefab folder.

Move VideoManager to the Hierarchy browser.

Select VideoManager to display the options in the Inspector window.

The options are described below.

i.   **StrFileName:** Enter the name of the file to play.

    1.   Enter the general file name in the case of a StreamingAssets video.

    2.   Enter the absolute path, such as file:///sdcard/test.mp4, in the case of a video stored in SD card.

    3.   Enter the URL, such as http://www.test.com/test.mp4, in the case of a streaming video.

ii.   **Target Material**: Link the GameObject to be replaced with video texture. (In the example,

    VideoManager itself was linked.) iii. **BFull Screen**: Select it only in the case of a full screen video. (It is

selected only in the case of FullScreen Prefab. Deselect it in other cases.)

iv.   **BSupport Rockchip**: In the case of Rockchip chipset, only 16 bit video buffer is supported. Also, in the case of Rockchip, Asset videos (in the StreamingAssets folder) cannot be played properly when accessed directly. This option is used to solve these problems. (The video quality may suffer slightly when this option is used.)

v.   **ScaleValue**: Determines which axis to use to resize the game object.

vi.   **objResize**: Determines the game object to resize.

   If this option is null, no action is taken.

vii.   **bLoop**: Determines whether to replay the video automatically when the video is over.

viii.   **bAutoPlay**: Determines whether to play the game object automatically when it is activated.

Using Basic Functions

1. void Load(string strFileName);

➔   Enter the name of the file or URL to load.

➔   If another video is being played, this function unloads the existing video and loads the new video automatically.

➔   You can enter the file name or URL as follows:

   ■   Enter the general file name in the case of a StreamingAssets video.

   ■   Enter the absolute path, such as file:///sdcard/test.mp4, in the case of a video stored in SD card.

   ■   Enter the URL, such as http://www.test.com/test.mp4, in the case of a streaming video.

2. void Play();   ➔ Plays the video.

➔ This function plays the video when it is called at the Stop, Pause or Ready state.

3. void Stop();

➔   Stops the video.

➔   When you stop the video by calling Stop() and play it again, the video is played from the beginning.

4. void Pause();

➔   Pauses the video.

➔   When you pause the video by calling Pause() and play it again, the video is played from where it was paused.

5. void UnLoad();

➔ Unloads the current video from the memory.

➔ You can call this function regardless of the current state.

6. int GetDuration();

➔ Fetches the total length of the current video.

➔ The length is in milliseconds.

➔ If it is a streaming video, the function returns -1.

➔ In the case of a streaming video, you need to use the GetCurrentSeekPercent() function.

7. int GetCurrentSeekPercent(); => Only Android (Does not work in iOS)

➔ Fetches the buffering degree of a streaming video.

➔ The function performs as follows according to the MediaPlayer API:

Get update status in buffering a media stream received through progressive HTTP download. The received buffering percentage indicates how much of the content has been buffered or played.

For example a buffering update of 80 percent when half the content has already been played indicates that the next 30 percent of the content to play has been buffered.

the percentage (0-100) of the content that has been buffered or played thus far

8. int GetVideoWidth(), int GetVideoHeight ()

➔ Fetches the width and height of the current video respectively.

9. void SeekTo(int iSeek)

➔ Change the video playback position. (ms units)

10. int GetSeekPosition()

➔ Gets the current playback position. (ms units)

11. Void SetVolume(float fVolume) Adjust the volume. (0~1.0)

12. void SetSpeed(float fSpeed) Adjust the video play speed.. (0 ~ 3.0) ,
(PC&iOS support, support from Android version 6.0 or later)

13.MEDIAPLAYER_STATE GetCurrentState() It gets the value of the current player state.

```
public enum MEDIAPLAYER_STATE
{
        NOT_READY = 0,
        READY = 1,
        END = 2,
        PLAYING = 3,
```

```
        PAUSED = 4,
        STOPPED = 5,
        ERROR = 6
}
```
14. DeleteVideoTexture() After the play ends, it used to delete the last flame cache.

In general, do not use.

Event basic usage

Support for the following events.

```
public VideoResize OnResize;    // When the video size is changed
public VideoReady OnReady;      // When the video is ready to use
public VideoEnd OnEnd;          // When the video finishes playing
public VideoError OnVideoError;// When an error occurs
public VideoFirstFrameReady OnVideoFirstFrameReady; // When the first screen
of the video is ready
```


```
public class MediaPlayerEvent : MonoBehaviour {


    public MediaPlayerCtrl m_srcVideo;

    // Use this for initialization
    void Start () {
        m_srcVideo.OnReady += OnReady;
        m_srcVideo.OnVideoFirstFrameReady += OnFirstFrameReady;
        m_srcVideo.OnVideoError += OnError;
        m_srcVideo.OnEnd += OnEnd;
        m_srcVideo.OnResize += OnResize;



    }

    // Update is called once per frame
    void Update () {

    }

    void OnReady() {

        Debug.Log ("OnReady");
```

```
    }

    void OnFirstFrameReady() {
        Debug.Log ("OnFirstFrameReady");
    }

    void OnEnd() {
        Debug.Log ("OnEnd");
    }

    void OnResize()
    {
        Debug.Log ("OnResize");
    }

    void OnError(MediaPlayerCtrl.MEDIAPLAYER_ERROR errorCode, MediaPl
ayerCtrl.MEDIAPLAYER_ERROR errorCodeExtra){
        Debug.Log ("OnError");
    }
```

Notes

1. The software for Android uses MediaPlayer class of Android. It does not use open source codes such as ffmpeg.
2. You need Unity Pro for iOS. (In the case of Unity 4.X, Unity 5.X does not need the Pro version.)
3. If dll error, install the packages below. And install the latest graphics drivers.
    https://www.microsoft.com/en-us/download/details.aspx?id=48145
4. The software requires Android version 4.0 or above.
5. Multi-thread rendering option is supported in unity 5.0