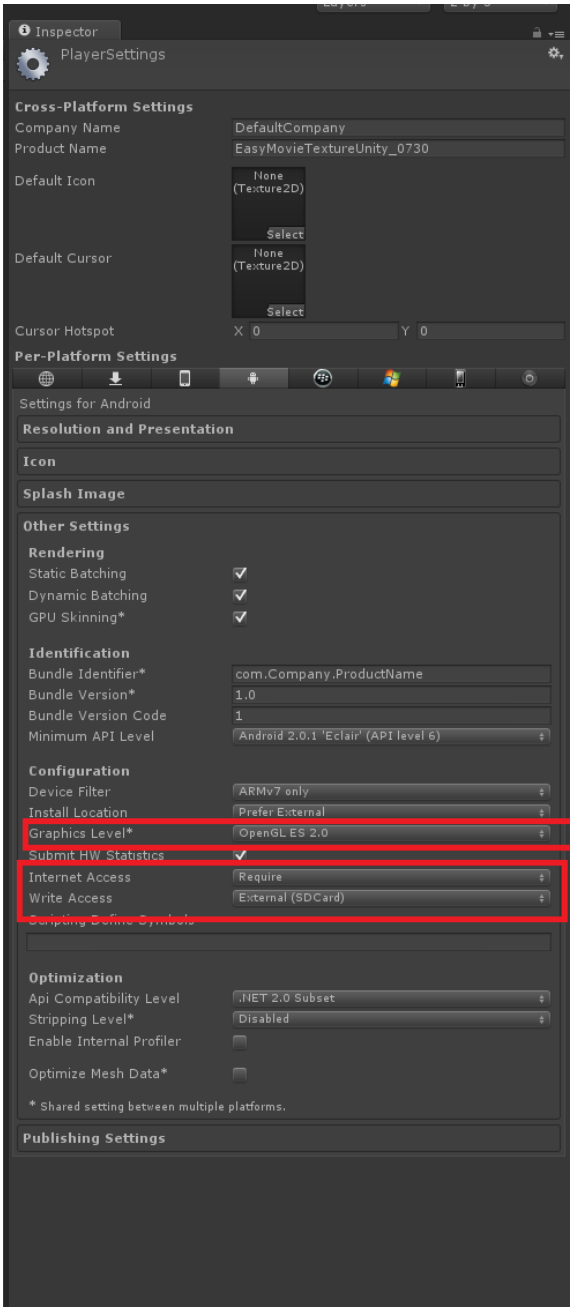
EasyMovieTexture Manual For Android

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)



Multi-thread rendering option is supported in Unity 4.3 or above.

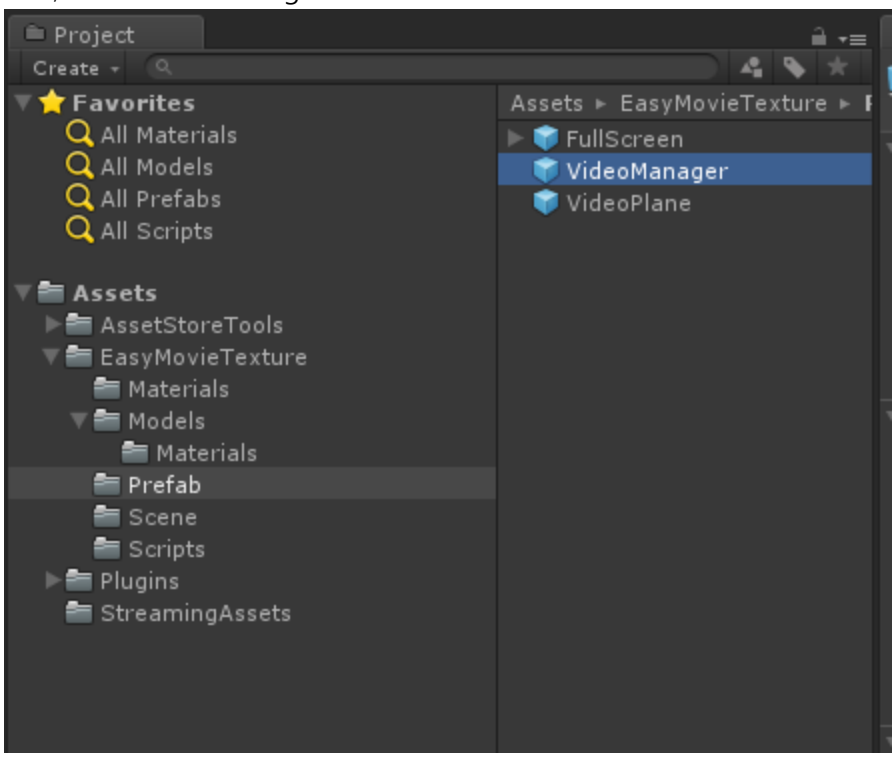
Currently, however, EasyMovieTexture does not support multi-thread rendering.

Therefore you need to deactivate the multi-thread rendering option.

In the case of iOS, Unity 4.6.3 or higher, Unity 5.0 or higher, another of the patch is required. EasyMovieTexture folder Unity463_Patch_IOS, there is Unity5_Patch_IOS. Please be patch to match the version.

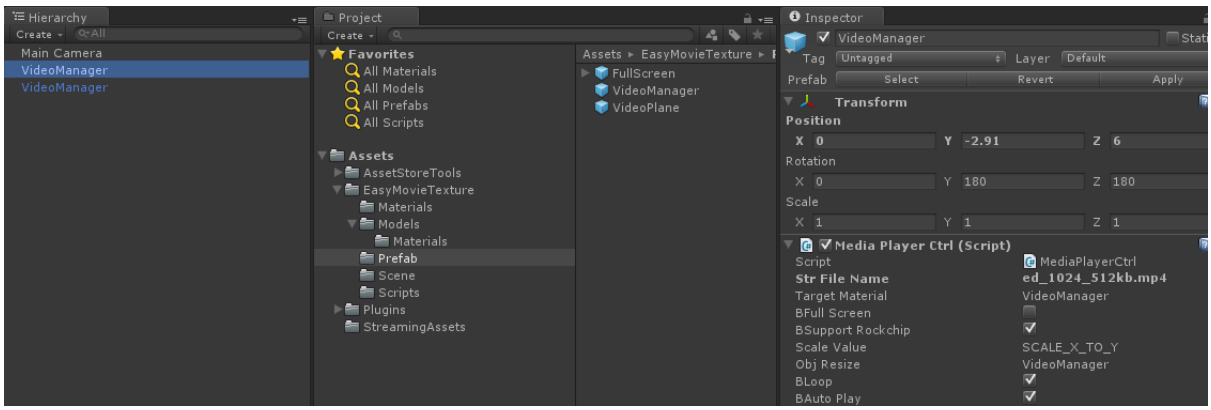
Using VideoManager

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.

Demo

Included demo videos are as follows:

1. Demo: Video demonstrating how to use the player.

2. Demo_FullScreen: Demo video of a full-screen format.
3. Demo_Sphere: A video demonstration using Sphere.

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();

➔ 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)

Notes

1. The software for iOS is a beta version and is still under development.

The iOS version of the software was made by modifying the source in https://github.com/unity3d-jp/iOS-VideoPlayerPlugin, which was released by the Unity subsidiary in Japan.

2. The software for Android uses MediaPlayer class of Android. It does not use open source codes such as ffmpeg.

3. You need Unity Pro for iOS.

4. The software does not function in the Editor mode. It functions in the device of respective platform.

5. The software requires Android version 4.0 or above.

6. Multi-thread rendering is not supported.