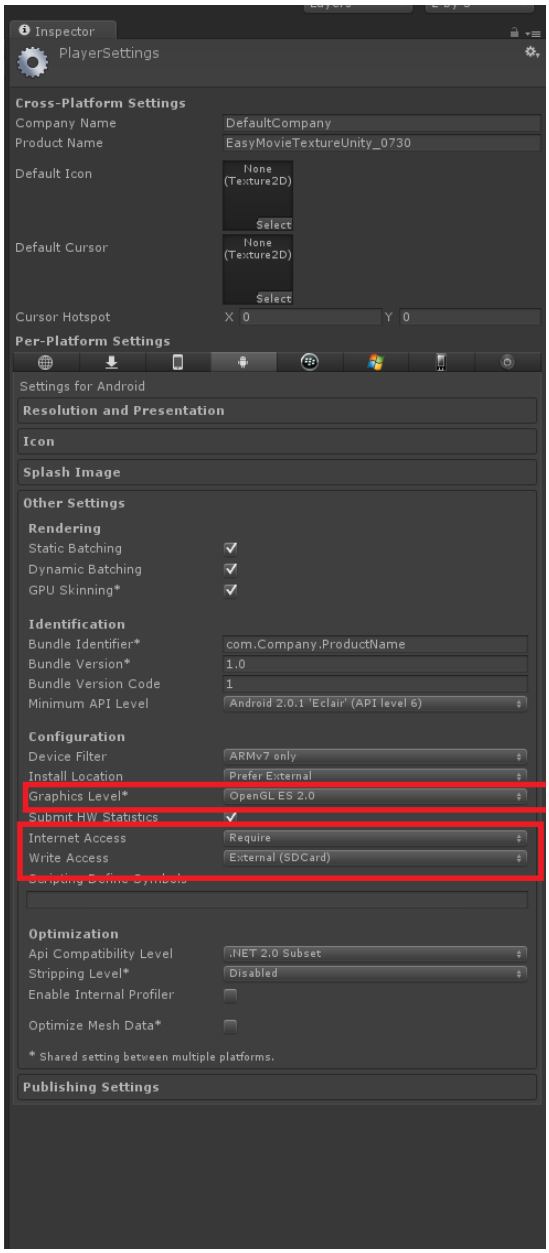
EasyMovieTexture For Android 매뉴얼

Project Setting

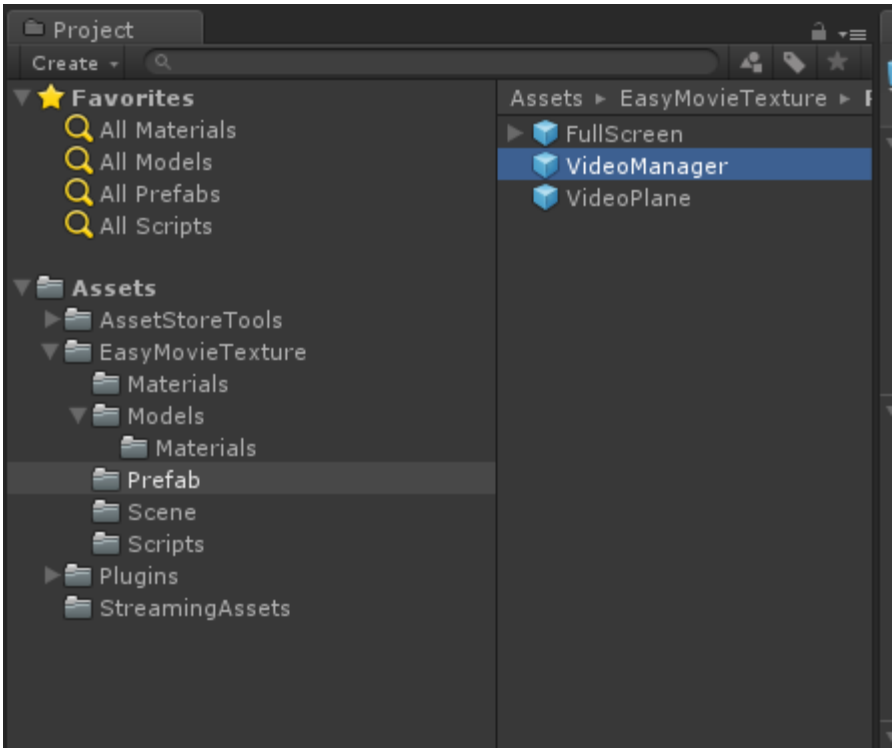
먼저 스트리밍 서비스와 sdcard의 영상을 사용하기 위해서는 아래와 같이 셋팅 하여야 합니다.
(File->BuildSettings->PlayerSettings->OtherSettings)



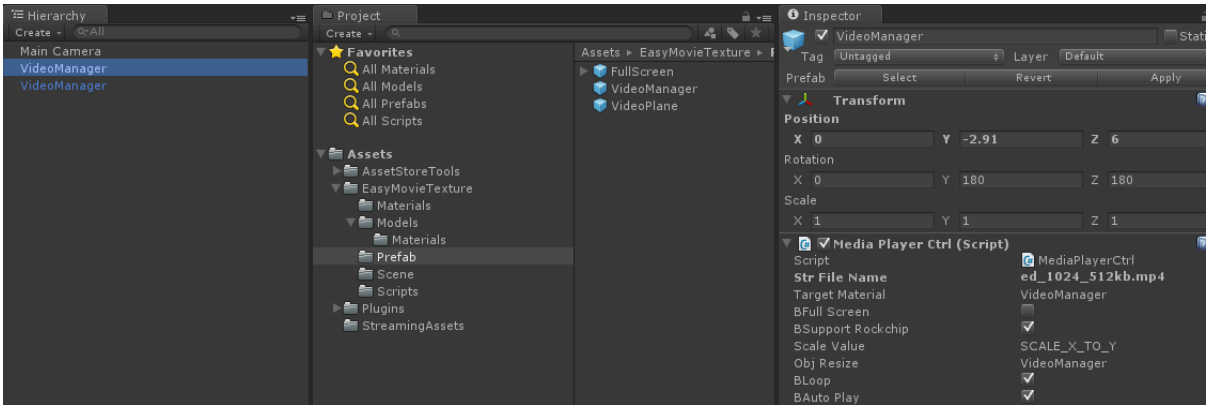
iOS 의 경우 Unity 버전 4.6.3 이상이거나 또는 Unity 5.0 이상라면 별도의 패치를 해주셔야합니다. EasyMovieTexture 폴더안에 Unity463_Patch_IOS, Unity5_Patch_IOS 가 있습니다. 버전에 맞게 패치 해주세요.

VideoManger 사용

먼저 Prefab폴더의 VideoManager를 선택합니다.



선택한 VideoManager를 Hierarchy Browser로 이동시킵니다.
이동된 VideoManager를 선택하면 Inspector 창에 옵션들이 보입니다.



해당의 옵션의 설명은 다음과 같습니다.

i. **StrFileName** : 재생하려는 파일명을 입력하시면 됩니다.
   1. StreamingAssets의 영상의 경우 일반적인 파일명을 적으시면 됩니다.
   2. SDCard의 영상의 경우 file:///sdcard/test.mp4 같이 절대경로로 입력하시면 됩니다.
   3. Streaming 영상의 경우 http://www.test.com/test.mp4 같이 URL 을 입력하시면 됩니다.

ii. **Target Material** : 영상 Texture로 교체하려는 GameObject를 연결시켜 줍니다.(여기서는 자기자신을 연결시켜 놓았습니다.)

iii. **BFull Screen** : 풀스크린 영상의 경우에만 사용합니다. (FullScreen Prefab에서만 체크되어있고 나머지경우 체크를 해제 해주시기 바랍니다.)

iv. **BSupport Rockchip** : Rockchip의 칩셋의 경우 비디오 버퍼가 16bit 밖에 지원이 안되는 문제가 있습니다. 또한 Rockchip의 경우 Asset[StreamingAssets 폴더]의 영상을 직접적으로 접근하여 플레이 할 경우 정상적으로 플레이가 안되는 문제가 있습니다. 그 부분을 해결하기 위한 코드의 사용여부 입니다. (사용시 퀄리티가 약간 떨어질수 있습니다.)

v. **ScaleValue** : 어느 축을 기준으로 게임오브젝트를 리사이즈 할것인지 설정합니다.

vi. **objResize** : 어떤 게임 오브젝트를 리사이즈 할것인지 설정부분입니다.
   만약 null 이면 아무런 행동도 하지 않습니다.

vii. **bLoop** : 영상이 끝날 때 자동 반복 재생을 할지 여부입니다.

viii. **bAutoPlay** : 게임오브젝트가 활성화 될 때 자동으로 플레이 할지 여부입니다.

포함된 데모

포함된 데모는 아래와 같습니다.

1. Demo : 일반적인 사용법에 대한 데모입니다.
2. Demo_FullScreen : 풀스크린 형식으로 사용하는 데모입니다.
3. Demo_Sphere : 구를 사용하는 데모입니다.

기본적인 함수 사용법

1. void Load(string strFileName);
   - ➔ Load하려는 파일명 또는 URL 을 입력하시면 됩니다.
   - ➔ 기본적으로 다른영상이 재생중이면 자동으로 플레이중인 영상을 UnLoad 시킨후 Load합니다.
   - ➔ 파일명 또는 URL 은 다음과 같이 입력하시면 됩니다.
     - ■ StreamingAssets의 영상의 경우 일반적인 파일명을 적으시면 됩니다.
     - ■ SDCard의 영상의 경우 file:///sdcard/test.mp4 같이 절대경로로 입력하시면 됩니다.
     - ■ Streaming 영상의 경우 http://www.test.com/test.mp4 같이 URL 을 입력하시면 됩니다.
2. void Play();
   - ➔ 영상을 재생합니다.
   - ➔ 일반적으로 Stop 또는 Pause 또는 Ready 상태에서 호출시 영상이 재생됩니다.

3. void Stop();
   - ➔ 영상을 멈춥니다.
   - ➔ Stop() 호출 후 플레이시 처음부터 플레이 됩니다.

4. void Pause();
   - ➔ 영상을 일시적으로 멈춥니다.
   - ➔ Pause() 호출 후 플레이시 멈춘 부분부터 플레이 됩니다.

5. void UnLoad();
   - ➔ 현재 영상을 메모리에서 제거 합니다.
   - ➔ 현재 상태에 상관없이 호출하시면됩니다.

6. int GetDuration();
   - ➔ 현재 영상의 총길이를 가져 옵니다.

➔ Milliseconds 단위로 가져옵니다.
➔ 스트리밍 서비스의 경우 -1을 리턴합니다.
➔ 스트리밍 서비스의 경우 GetCurrentSeekPercent() 함수를 사용하여야 합니다.

7. int GetCurrentSeekPercent(); (안드로이드 전용 ,iOS에서는 동작하지 않습니다.)
   ➔ 스트리밍 서비스의 버퍼링 정도를 나타냅니다.
   ➔ MediaPlayer API 에 따르면 다음과 같은 동작을 합니다.

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 ()
   ➔ 각각 현재의 영상의 너비와 높이를 가져옵니다.
9. void SeekTo(int iSeek)
   영상 재생 위치를 변경합니다. ms 단위 입니다.

참고사항

1. 현재 iOS의 경우 Beta Version으로 아직 기능 수정 중에 있습니다.
iOS의 경우 일본 Unity 지사에서 공개한 https://github.com/unity3d-jp/iOS-VideoPlayerPlugin 소스를 수정하여 만들었습니다.

2. Android는 Android의 MediaPlayer class 를 사용합니다. ffmpeg같은 오픈소스는 사용하지 않습니다.

3. iOS 의 경우 Unity Pro 가 요구됩니다. (Unity 4.X 의 경우, Unity 5.X에서는 Pro 버전이 필요하지 않습니다.)

4. Editor에서는 동작하지 않습니다. 각 플랫폼 디바이스에서 동작합니다.

5. Android 버전 4.0 이상을 요구합니다.

6. 멀티쓰래드 랜더링은 유니티 5.0 부터 지원합니다.