EasyMovieTexture For Android 매뉴얼

포멧 및 해상도

지원포멧 :

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

지원해상도:

Android: 일반 디바이스의 경우 1920*1080까지 지원합니다.

최신 디바이스의 경우 4k 까지 지원합니다.

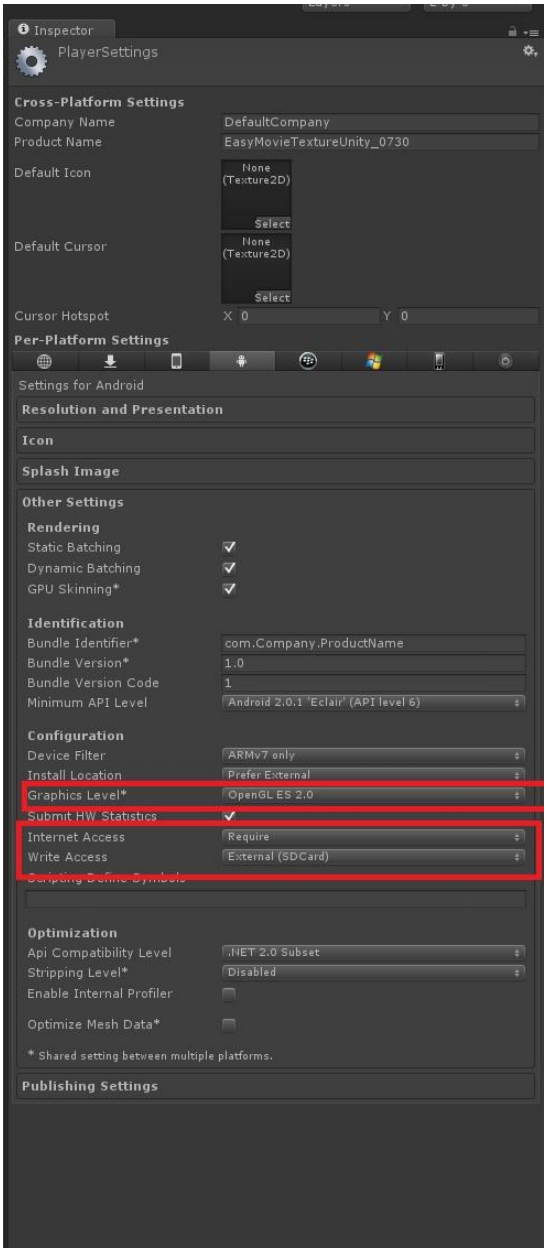
iOS: 일반디바이스의 경우 1920*1080까지 지원합니다.

최신 디바이스의 경우 2560 * 1440 까지 지원합니다..

4k해상도의 경우 아이폰 6s plus 이상을 지원합니다.

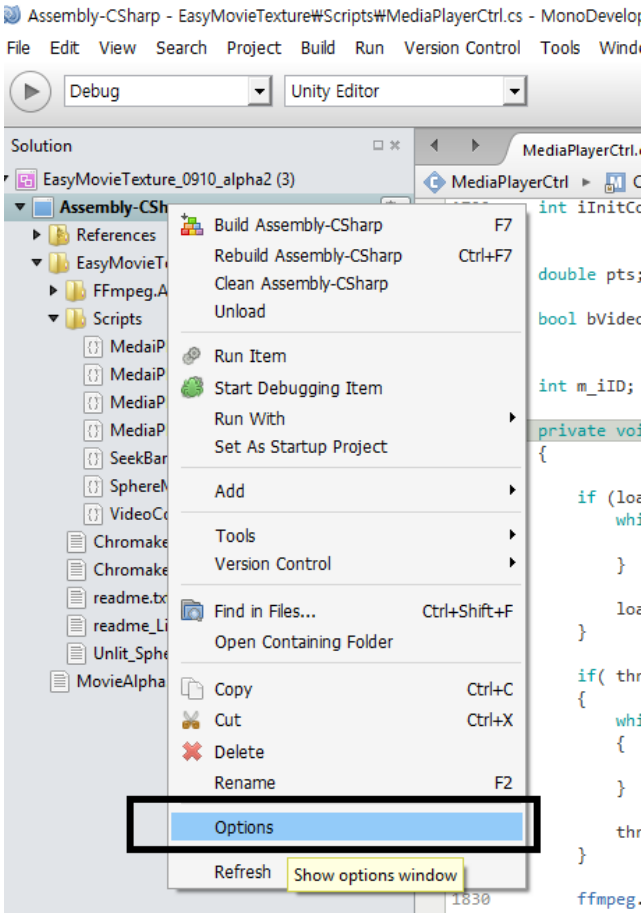
Project Setting

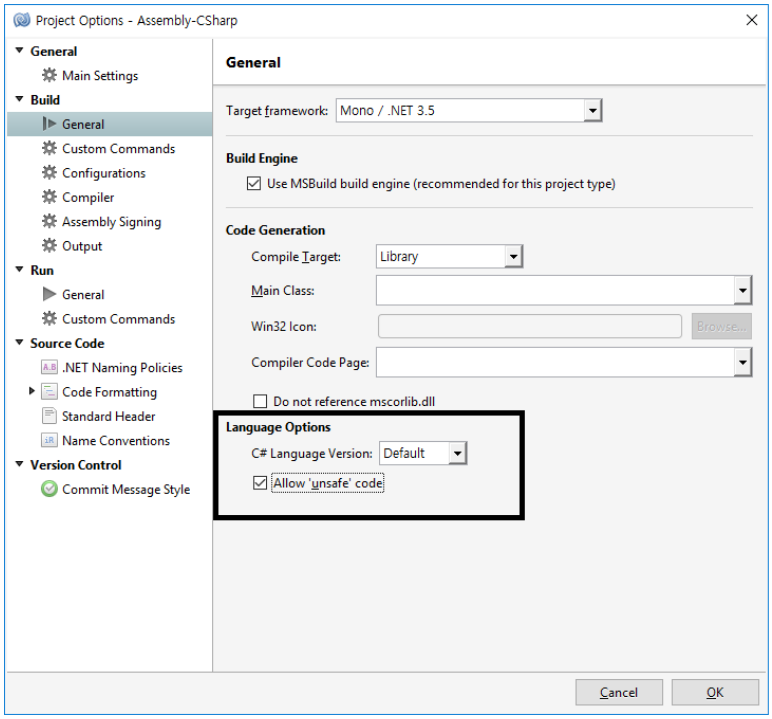
먼저 스트리밍 서비스와 sdcard 의 영상을 사용하기 위해서는 아래와 같이 셋팅 하여야 합니다.
(File->BuildSettings->PlayerSettings->OtherSettings)



iOS 의 경우 Unity 버전 4.X 의경우, EasyMovieTexture 폴더안에 Unity463_Patch_IOS 가 있습니다. 버전에 맞게 패치 해주세요.

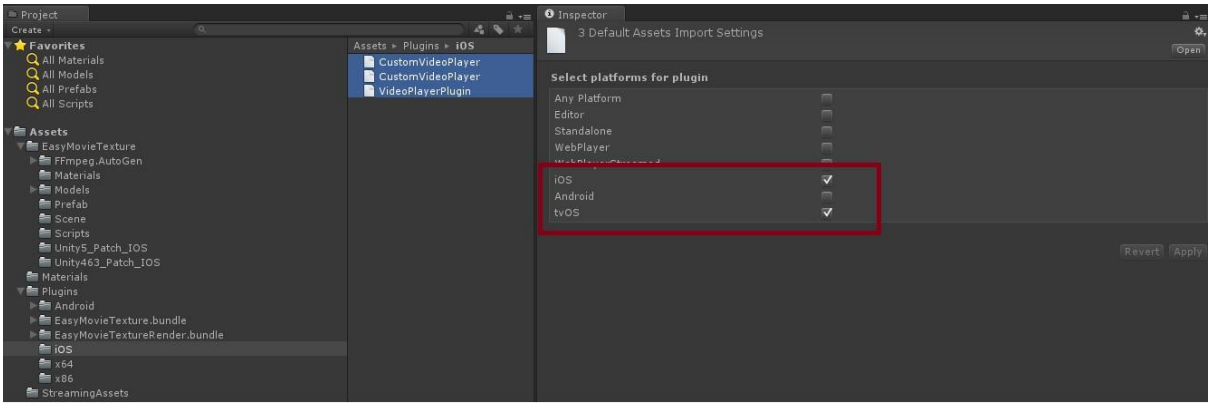
디버그 모드 사용시 아래와 같이 사용하세요.


Assembly-CSharp - EasyMovieTexture₩Scripts₩MediaPlayerCtrl.cs - MonoDevelop
File Edit View Search Project Build Run Version Control Tools Wind
Debug
Unity Editor
Solution
EasyMovieTexture_0910_alpha2 (3)
References
Scripts
Build Assembly-CSharp F7
Rebuild Assembly-CSharp Ctrl+F7
Clean Assembly-CSharp
Unload
Run Item
Start Debugging Item
Run With
Set As Startup Project
Add
Tools
Version Control
Find in Files... Ctrl+Shift+F
Open Containing Folder
Copy Ctrl+C
Cut Ctrl+X
Delete
Rename F2
Options
Refresh
Show options window
MediaPlayerCtrl
int iInitCo
double pts;
bool bVidec
int m_iID;
private voi
{
if (loa
whi
}
loa
}
if(thr
{
whi
{
}
thr
}
ffmpeg.
1830


Project Options - Assembly-CSharp
General
Main Settings
Build
General
Custom Commands
Configurations
Compiler
Assembly Signing
Output
Run
General
Custom Commands
Source Code
.NET Naming Policies
Code Formatting
Standard Header
Name Conventions
Version Control
Commit Message Style
General
Target framework: Mono / .NET 3.5
Build Engine
Use MSBuild build engine (recommended for this project type)
Code Generation
Compile Target: Library
Main Class:
Win32 Icon:
Browse...
Compiler Code Page:
Do not reference mscorlib.dll
Language Options
C# Language Version: Default
Allow 'unsafe' code
Cancel
OK

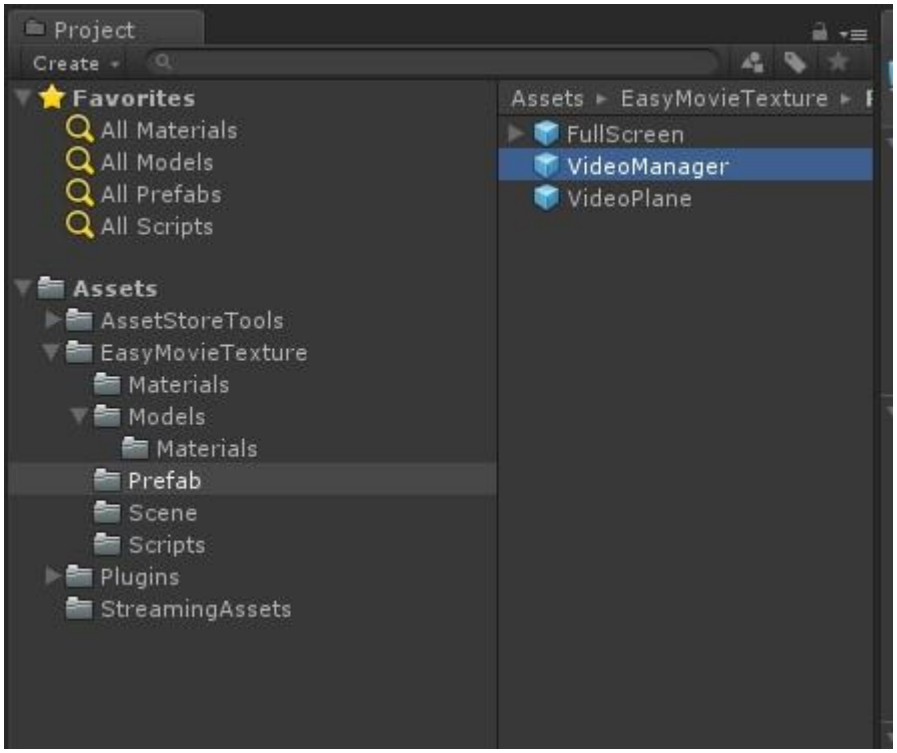
AppleTV (tvOS) 사용시 설정

1. 아래 이미지와 같이 설정해주세요.



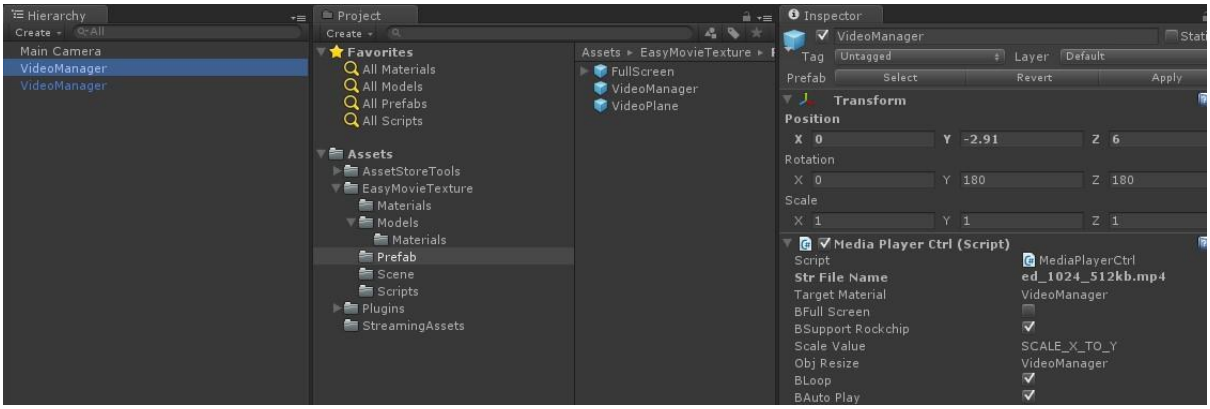
VideoManger 사용

먼저 Prefab 폴더의 VideoManager 를 선택합니다.



선택한 VideoManager 를 Hierarchy Browser 로 이동시킵니다.

이동된 VideoManager 를 선택하면 Inspector 창에 옵션들이 보입니다.



해당의 옵션의 설명은 다음과 같습니다.

i. **StrFileName** : 재생하려는 파일명을 입력하시면 됩니다.

1. StreamingAssets 의 영상의 경우 일반적인 파일명을 적으시면 됩니다.
2. SDCard 의 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 Texture 로 교체하려는 GameObject 를 연결시켜 줍니다.(여기서는 자기자신을 연결시켜 놓았습니다.)

iii. **BFull Screen** : 풀스크린 영상의 경우에만 사용합니다. (FullScreen Prefab 에서만 체크되어있 고 나머지경우 체크를 해제 해주시기 바랍니다.)

iv. **BSupport Rockchip** : Rockchip 의 칩셋의 경우 비디오 버퍼가 16bit 밖에 지원이 안되는 문 제가 있습니다. 또한 Rockchip 의 경우 Asset[StreamingAssets 폴더]의 영상을 직접적으로 접근하여 플레이 할 경우 정상적으로 플레이가 안되는 문제가 있습니다. 그 부분을 해결 하기 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.

기본적인 함수 사용법

1. void Load(string strFileName);
   - ➔ Load 하려는 파일명 또는 URL 을 입력하시면 됩니다.
   - ➔ 기본적으로 다른영상이 재생중이면 자동으로 플레이중인 영상을 UnLoad 시킨후 Load 합니다.
   - ➔ 파일명 또는 URL 은 다음과 같이 입력하시면 됩니다.
     - ■ StreamingAssets 의 영상의 경우 일반적인 파일명을 적으시면 됩니다.
     - ■ SDCard 의 영상의 경우 file:///sdcard/test.mp4 같이 절대경로로 입력하시면 됩니다.
     - ■ Streaming 영상의 경우 http://www.test.com/test.mp4 같이 URL 을 입력하시면 됩니 다.
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 -1 을 리턴합니다.
   - ➔ 스트리밍 서비스의 경우 GetCurrentSeekPercent() 함수를 사용하여야 합니다.

7. int GetCurrentSeekPercent(); (안드로이드 전용 ,iOS 에서는 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.
10. int GetSeekPosition() 현재 영상 재생 위치를 가져옵니다. Ms 단위 입니다.
11. Void SetVolume(float fVolume) 볼륨을 조절합니다. (0~1.0)
12. SetSpeed() 영상 재생 스피드를 조절합니다. (0 ~ 3.0) ,안드로이드의 경우 6.0 이상부터 동작합니다.
13. GetCurrentState() 현재 플레이어 상태 값을 가져옵니다. 상태값은 아래와 같습니다.

```
public enum MEDIAPLAYER_STATE
{
        NOT_READY = 0,
        READY = 1,
        END = 2,
        PLAYING = 3,
        PAUSED = 4,
        STOPPED = 5,
        ERROR = 6
}
```

14. DeleteVideoTexture() 플레이 종료후, 마지막 프레임 캐시를 삭제할 때 사용합니다.

일반적으로는 사용하지 마세요.

기본적인 이벤트 사용법

기본적으로 아래와 같은 이벤트가 있습니다.

```
public VideoResize OnResize;   //비디오 사이즈가 변경될때
public VideoReady OnReady;     //비디오 사용이 준비되었을때
public VideoEnd OnEnd;         //비디오가 끝났을때
public VideoError OnVideoError;//에러 발생시
public VideoFirstFrameReady OnVideoFirstFrameReady; //비디오의 첫화면이 준비되었을 때
```

```
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");
    }
}
```

참고사항

1. iOS 의 경우 Unity Pro 가 요구됩니다. (Unity 4.X 의 경우, Unity 5.X 에서는 Pro 버전이 필요하지 않습 니다.)

2. Android 버전 4.0 이상을 요구합니다.

3. 멀티쓰래드 랜더링은 유니티 5.0 부터 지원합니다.