UNIVERSAL MEDIA PLAYER

User Manual

Version 2.0.1

CONTENTS

USER SUPPORT

If you need support or have any questions or suggestions, please contact with me:

- unitydirectionkit@gmail.com (Main)
- unitydirectionkit@outlook.com (Additional)

INTRODUCTION

Universal Media Player (UMP) is cross platform media framework plugin for Unity that based on VLC and FFmpeg native libraries:

Platforms	CPUs	Library	Checked platforms/Graphic API
Windows	x86/x86_64	VLC	Windows 7+ (D3D9, D3D11, OpenGL)
OSX	x86_x64	VLC	10.10 Yosemite+ (OpenGL)
Linux	x86/x86_64/Universal	VLC	Ubuntu 12.04.5 LTS+ (OpenGL)
Android	armeabi-v7a/x86	VLC/Native	Android API level 14+ (Android 4.0+) (OpenGLES 2.0 or 3.0)
iOS	arm64, armv7	FFmpeg/Native	iOS 6+ (OpenGLES 2.0 or Metal)
WebGL (experimental)	x86/x86_64	HTML5 Video	Firefox, Chrome, Opera

Possibilities of current version:

- Fast and flexible video playback (fast native texture updates: **Direct3D9**, **Direct3D11** and **OpenGL**);
- Fully compatible with Unity Editor (in-editor playback for Windows, OSX and Linux platforms);
- Supported Unity "Audio Source" component (works only on Windows, OSX and Linux platforms);
- Supported possibility to use external libVLC libraries (works only on Windows, OSX and Linux platforms);
- Supported possibility to easy switch subtitles (SPU) and audio tracks (works only on Windows, OSX and Linux platforms);
- Supported videos with transparent channel (works only on Windows, OSX and Linux platforms);
- Supported possibility to get pixels (snapshot) of playback video frame;
- Supported "**Youtube**" video playback;
- Supported main video file formats playback: 3GPP, AVI, FLV, SWF, M4V, Matroska, Ogg Video, QuickTime File Format, WebM, Windows Media Video and streaming media protocols: HTTPS, HTTP, HLS, RTSP, RTMP (if you have some additional stream or file format, please write me on my support email and I will check it for you, before you buy my asset:);
- Supported main video player events system: **Opening**, **Buffering**, **Prepared**, **Playing**, **Paused**, **Stopped**, **Ended**, **Error**;

- Supported full logging system from native library in Unity Editor for more debugging possibility with different depth: Warning, Debug, Error (works only on Windows, OSX platforms);
- Supported main video player features, like: play, pause, mute, playback rate, rewind, snapshot and other.

IMPORTANT

Please note that this package contains pre-compiled binaries for libvlc which are licensed under LGPL. Also this package contains not all of the libVLC modules, because some of them are licensed under GPL, so they has removed.

VLC native library source code used in this package is available: Winsows, OSX, Linux.

INSTALLATION GUIDELINE

Import the UMP package from the Asset Store. You should now have a folder named "UniversalMediaPlayer" with the following structure in your Unity project:

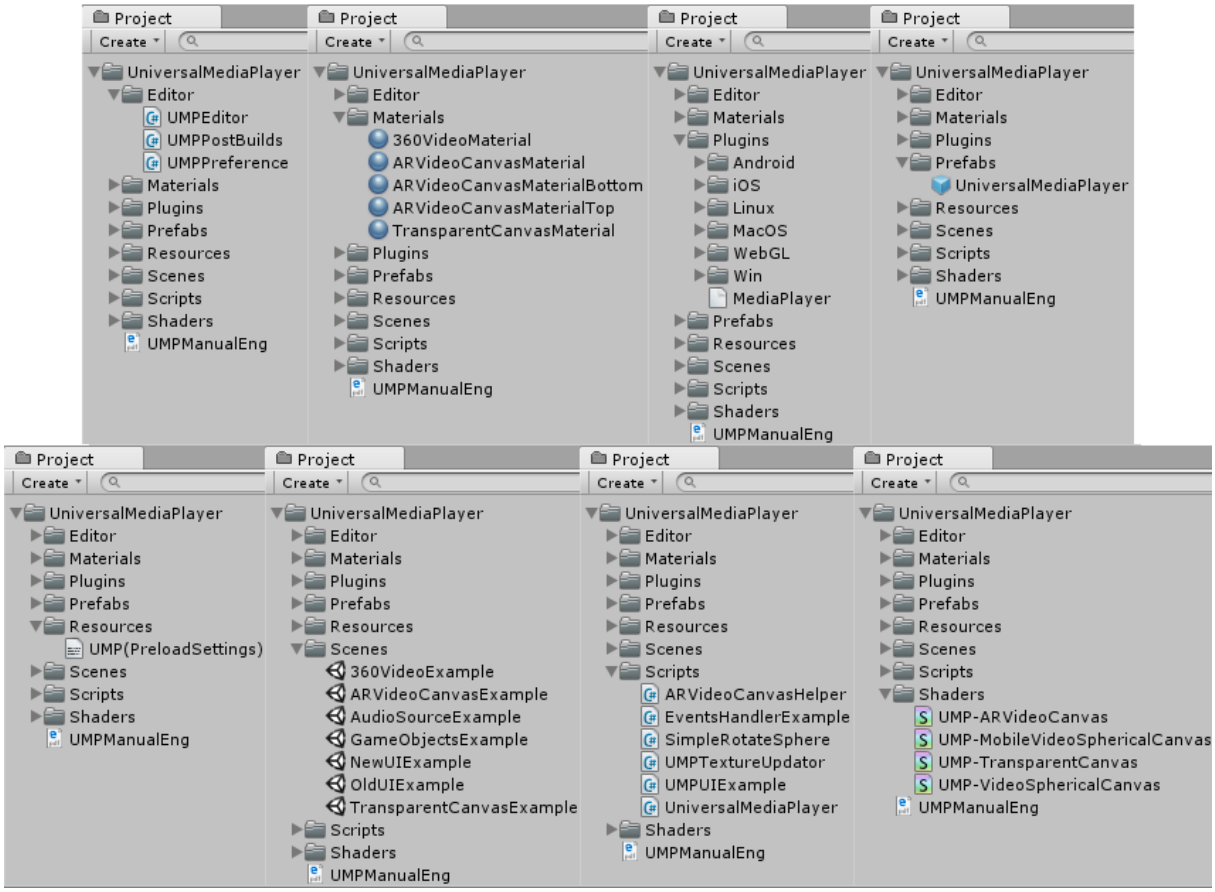


Figure 3.1: Structure of UMP asset

In short:

- **Editor**: Custom Editor, Preferences and Build scripts;
- **Materials**: Custom materials that used in example scenes that based on custom shaders;
- **Plugins**: All the native dlls and shared libraries;
- **Prefabs**: Special UMP prefab that used for easy setup your project with UMP asset;
- **Resources**: Special file where UMP asset stored all preloaded setting information;
- **Scenes**: Demo scenes that's show all components in a ready setup and how work with some additional features of UMP asset;
- **Scripts**: C# classes that show how to work with UMP;
- **Shaders**: Custom shaders that can be used for dynamic aspect handling and for video with transparent channel.

Asset Parts

Full UMP asset consists of two independent parts:

- **UMP (Win, Mac, Linux, WebGL)** – for desktop platforms support;
- **UMP (Android, iOS)** – for mobile platforms support.

So, if you bought only UMP for mobile platforms it's doesn't work in standalone builds (PC, Mac, Linux & WebGL platforms), but you will have possibility to playback video in Unity Editor – more information in External VLC Player Support subitem. The same with UMP for desktop platforms – you will have possibility to play videos in Unity Editor and standalone builds (PC, Mac, Linux & WebGL platforms), but it's doesn't work on mobile (Android, iOS) platforms.

To correctly combine two independent parts into one full UMP asset you need:

1. Check if you have the last versions of UMP asset for desktop and mobile platforms;
2. At first, import **UMP (Android, iOS)** part into your project (press "All" and after it "Import" button in "Import Unity Package" window);
3. Import **UMP (Win, Mac, Linux, WebGL)** part into your project (press "All" and after it "Import" button in "Import Unity Package" window);
4. If you will use **UMP Pro versions**, you also need to import special additional package that should replace some classes in main "Source" folder, to do this just import UMPPatch_Pro(combine_mobile_desktop) which will be included in your UMP asset in main root folder "UniversalMediaPlayer" (press "All" and after it "Import" button in "Import Unity Package" window):

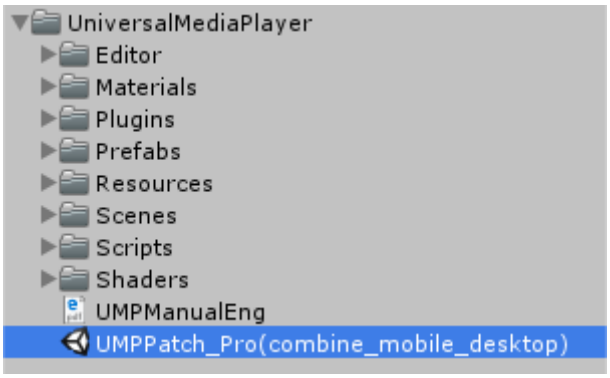


Figure 3.2: UMP patch to combine mobile and desktop versions

After all this steps you will have possibility to use full UMP features in one full asset.

External VLC Player Support

UMP asset has possibility to support external libVLC libraries that will be installed with regular VLC player. It's also give you possibility to playback video in Unity Editor witout internal libVLC libraries that included in UMP asset by default (for example if you don't want to sotore this

libraries in git repositories) and you can easily playback videos if you have only UMP (Android, iOS) part of full UMP asset.

To correctly use regular VLC player with UMP asset you need:

1. Download correct version of regular VLC player depending from Unity Editor bit version that you use:
    - Win Unity Editor (32bit)
    - Win Unity Editor (64bit)
    - Mac Unity Editor (64bit)
2. Check if all setup correctly in special UMP menu in Unity Preferences (more information in Preference Item chapter).

Updating Exist Version

When updating an existing UMP installation you should just delete the old "UniversalMediaPlayer" folder before importing the new version of UMP asset. If you had entered the play mode from Unity once in your editor session you have to close Unity and restart it. Otherwise the native dlls will be cached from Unity and could not be updated properly.

USAGE GUIDELINE

Preference Item

It's a special possibility to setup UMP asset with some additional settings. You can find this menu in:

- **Windows**: Edit->Preferences...->UMP
- **OSX**: Unity->Preferences...->UMP
- **Linux**: Edit->Preferences...->UMP

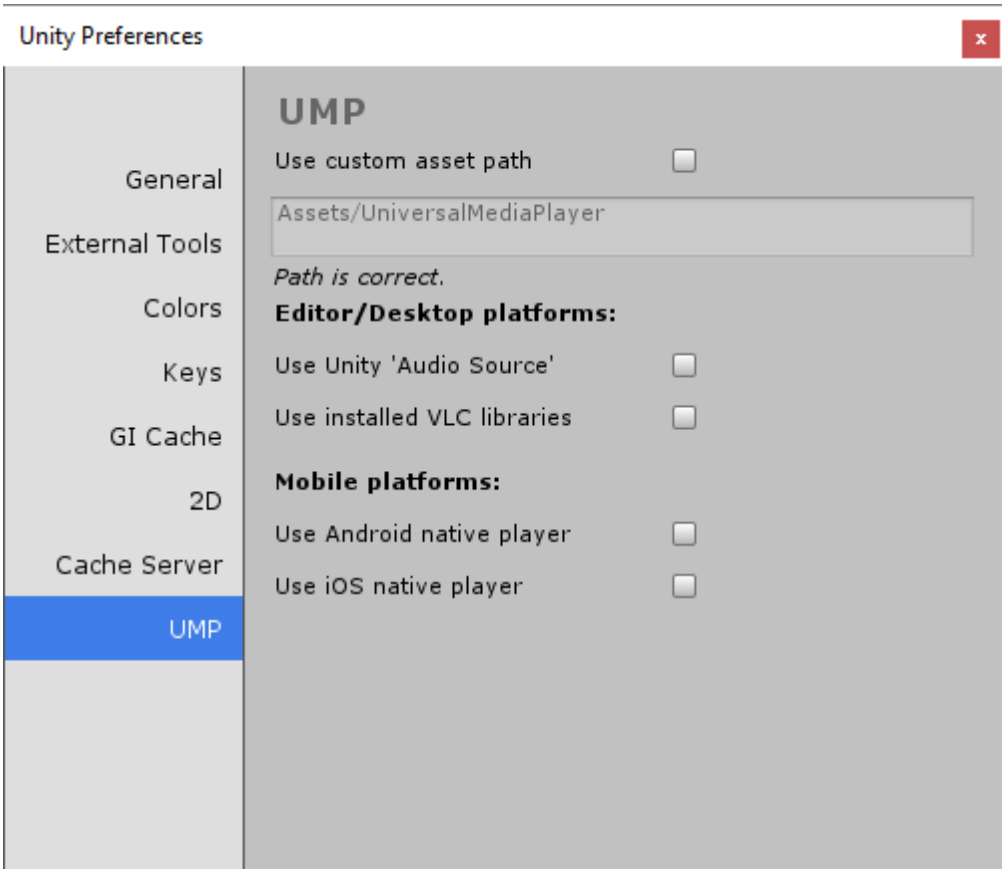


Figure 4.1: UMP additional settings

In short:

- **Use Unity 'Audio Source'**: will be using Unity 'Audio Source' component for audio output for all UMP instances (global) by default (works only on Windows, OSX and Linux platforms);
- **Use installed VLC libraries**: will be using external/installed VLC player libraries for all UMP instances (global, works only on Windows, OSX and Linux platforms). Path to install VLC directory will be obtained automatically (you can also setup your custom path):

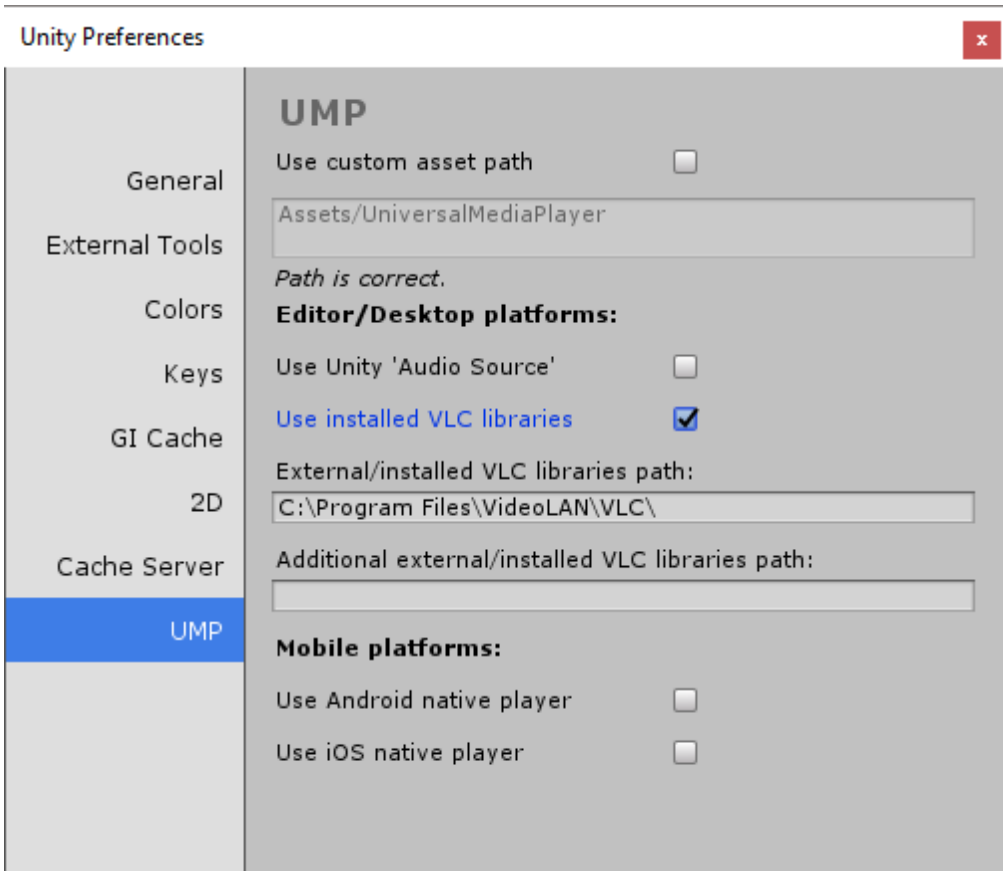


Figure 4.2: UMP "Use installed VLC libraries" setting

- **External/installed VLC libraries path**: Default path to installed VLC player libraries, it's directory will be obtained automatically if you will have installed regular VLC player;
- **Additional external/installed VLC libraries path**: Additional path to installed VLC player libraries. Will be used if path to libraries can't be automatically obtained.

- **Use Android native player**: will be using Android native media player for all UMP instances (global, works only for Android platform).
  - **Android native player**: has better compatibility with old devices and better support of playback video files from "StreamingAssets" folder. So, if your videos/stream work with this player it's better to use it by default;
  - **LibVLC player (by default)**: has better codecs support, but low devices support. So, if your videos/stream don't work with Android native player, use this one by default.
- **Use iOS native player**: will be using AVPlayer for all UMP instances (global, works only for iOS platform).

- **AVPlayer**: has better compatibility with old devices and take less space in builded version, but it's still experimental and will be updated as soon as possible;
- **FFmpeg player (by default)**: has better codecs support. So, if your videos/stream don't work with AVPlayer, use this one by default.

Prefab Usage Tutorial

You need to find the special UMP prefab in "Prefabs" folder that give you possibility to manage all media player functionality and move it to your Unity scene:

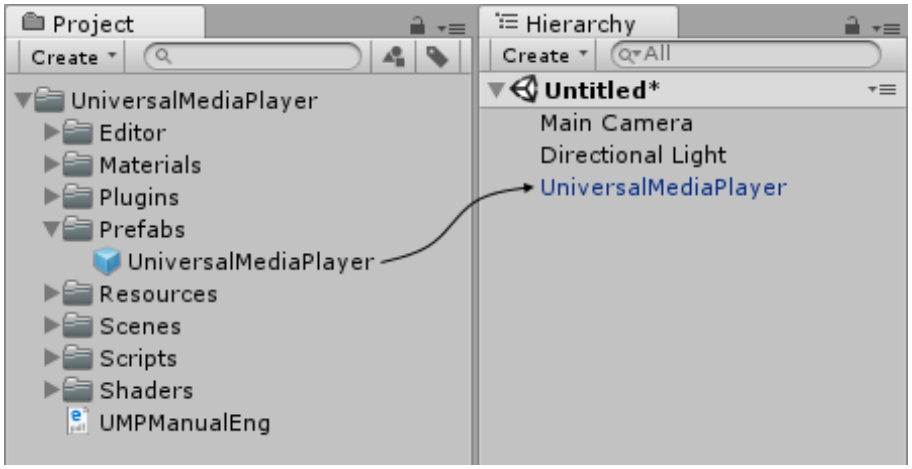


Figure 4.3: add new UMP instance

When you select the UMP instance in your "Hierarchy" window you can manage it with the component inspector:

Figure 4.4: UMP inspector view

In short:

- **Rendering GameObjects**: simple array that consist with Unity "GameObjects" that have "Mesh Renderer" with some material or "Raw Image" component:

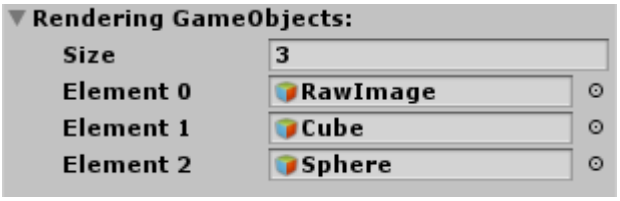


Figure 4.5: UMP rendering objects

- **Path to video file**: url link or local path to your video. Both of this will be work correctly and you don't need to worry about video file location. For local files you can specify an absolute path with the file:/// sheme on all supported platforms. For remote files or streams you just use your url link. Usage examples:
    - Play video from "StreamingAssets" folder (StreamingAssets\myVideoFile.mp4)- file:///myVideoFile.mp4;
    - Play video from local storage space (C:\MyFolder\Videos\video1.mp4) - file:///C:\MyFolder\Videos\video1.mp4 or C:\MyFolder\Videos\video1.mp4 or file:///DCIM/100ANDRO/MyVideo.mp4 (example for Android platform, so you can ignore root folder – in my case is "/storage/emulated/0/"). Also, if you want to play video on Android platform, don't forget to set 'Write Access' to 'External (SDCard) in your player settings;
    - Play video from remote space (play streams) - rtsp://wowzaec2demo.streamlock.net/vod/mp4:BigBuckBunny_115k.mov.
- **AutoPlay**: start playback automatically after video is buffered;
- **Looping**: When the playback reaches the end position it jumps to the start and plays again. Also, you have possibility to use "**Smooth Loop**":

Figure 4.6: UMP smooth loop

Differences between this features is:

- **Loop**: will be have buffering stage, but use lees devices resources;
- **Loop(smooth)**: don't have buffering stage, but use more devices resources.

- **Mute**: set mute status for current video playback;
- **Advanced options**: special properties that give you possibility to have additional setup for your video playback:

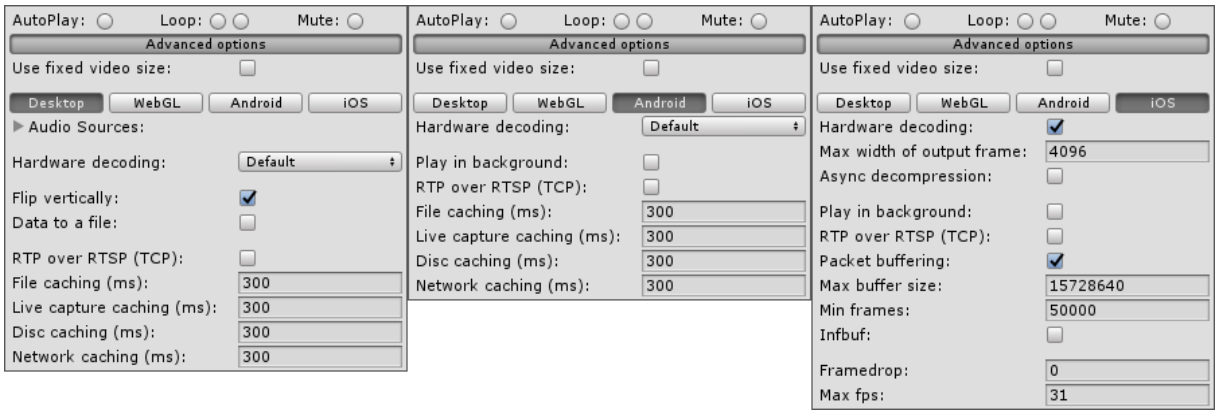


Figure 4.7: UMP advanced properties

So, if you will have not smooth playback (included some freezes) or you want to decrease buffering time, just try to use this additional properties, it's can help to solve your issue, aslo all this properties has special tooltips that explain what effect on video playback they has.

- **Volume**: set current software audio volume (by default you can change this value from "0" to "100");
- **Play rate**: set the requested movie play rate (by default you can change this value from "0.5" to "5", also it's don't support negative playback).
- **Position**: set movie position. This has no effect if playback is not enabled. This might not work depending on the underlying input format and protocol;
- **Load**: prepare video playback (wait, until we can't get first frame of current video);
- **Play**: start video playback;
- **Pause**: if a video is playing you can pause it. To continue playback you should press "Play" button again;
- **Stop**: stops the media and seeks to the first frame. You need to press "Play" button to start playback again;
- **Shot**: Take a snapshot of the current video playback and save it by default in "Application.persistentDataPath" folder (works only on Windows, OSX and Linux platforms);
- **Log Detail**: gives you possibility to getting log messages from native library (works only on Windows and OSX platforms):

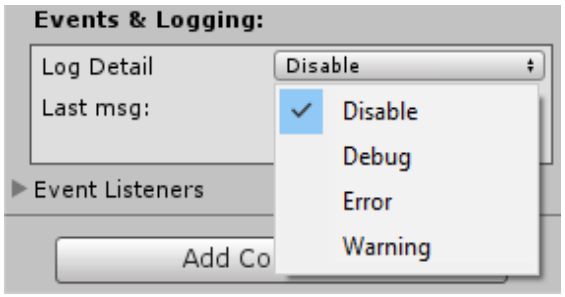


Figure 4.8: UMP log detail filters

- **Last msg**: display last playback event or playback error;
- **Event listeners**: gives possibility to attach to the special callbacks (Opening, Buffering, Playing, Paused, Stopped, End Reached, Encountered Error, Time Changed, Position Changed, Snapshot events):

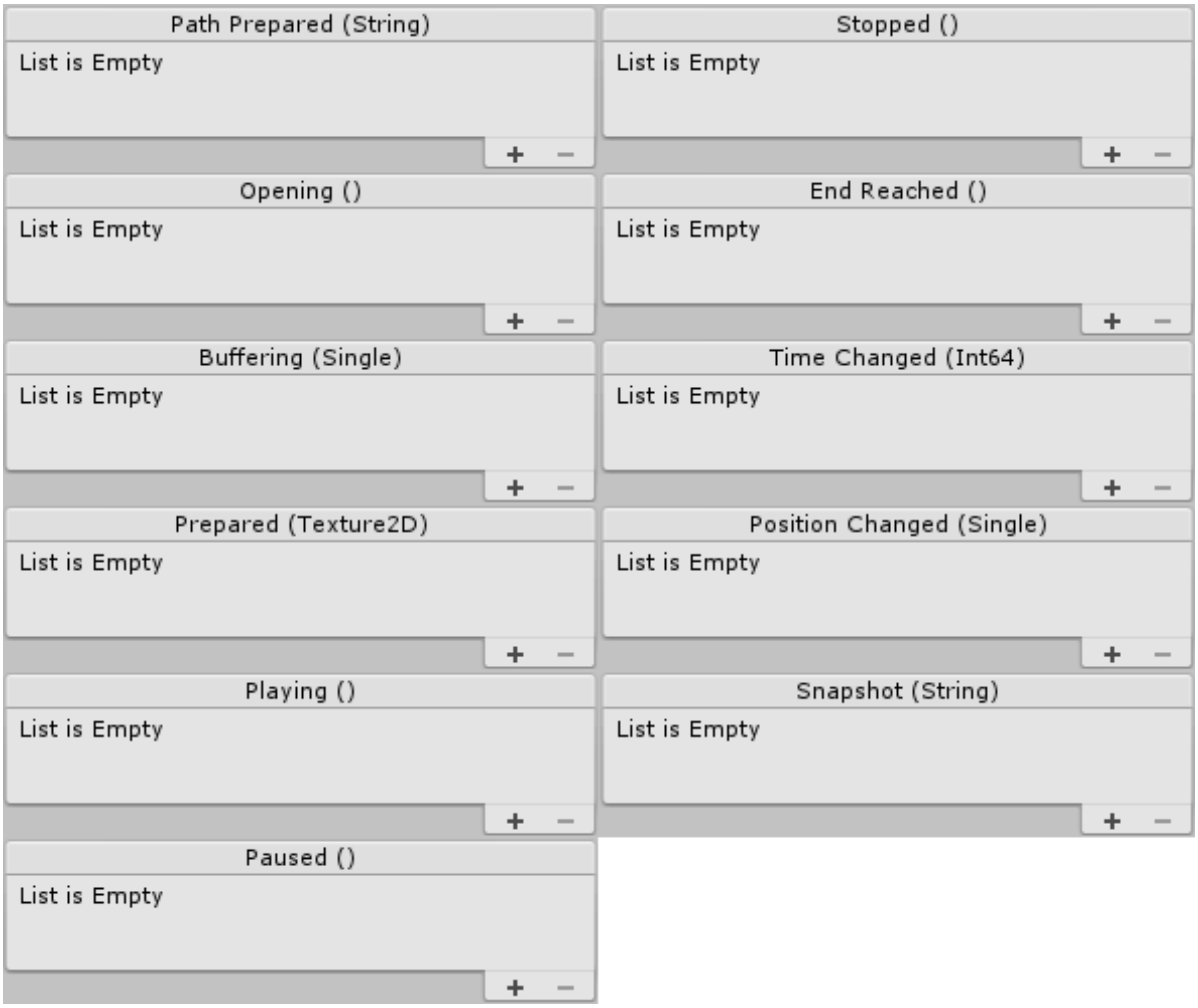


Figure 4.9: UMP log detail filters

Patches

If you want to try some additional functionality or use different version of using libraries, you can use special UMP patches (just import into your project that has UMP asset: press “All” and after it “Import” button in “Import Unity Package” window).

Android platform

By default UMP uses **libVLC 2.1.0 libraries for Android** platform, but different versions of libVLC libraries has useful fixed that can help you to resolve some issues:

LibVLC 2.0.6 - (better support of rtmp strems, better buffering system implementation);

LibVLC 2.1.12 / LibVLC 2.1.19 – (add subtitle support, fixed some issues with older versions);

iOS platform

If you want to load some specific video formats/streams on iOS platform or your video file/stream don’t work properly with default UMP framework you can try to use special patch, that contains new media framework that has all possible supported codecs and replace the old one that located in " UniversalMediaPlayer\Plugins\iOS" folder. This updated framework not contains by default, because he has very big size and in main cases is not needed, but you can download it separately from this link: Full framework patch

Also, if you what to use default framework, just download and import this patch: Default framework patch

Desktop patforms:

The newest versions of UMP asset for desktop platforms using the latest available stable versions of libVLC libraries, for Windosw it’s 2.2.6, for MacOS it’s 2.2.5 and for Linux it’s 2.2.1. But you have possibility to use latest “nightly builds” libVLC libraries, it’s can be unstable and sometime will be work worse than released stable libraries, but they contains many fixed issues with support some new codecs and other important implementations. So, if you want to use “nightly builds” libVLC libraries try to import this package: UMP "nightly builds" Libraries or use default one: UMP Stable Libraries.

Linux Platform Setup

INSTRUCTION FOR UNITY EDITOR AND STANDALONE BUILD

If you don't have installed regular VLC player (or don't wan’t install it) on your Linux OS, you need to make additional installation of some ffmpeg packages that gives you possibility to have correct video playback. Without this additional installation, UMP asset probably will not work properly on your Linux OS.

So, first of all, please add "Execute" permission to special UMP shell scripts "UMPInstaller.sh" and "UMPRemover.sh" - this scripts should be generated automatically when you complete building process of your Unity project also is available in “UniversalMediaPlayer\Plugins\Linux”

folder. They give you possibility to automate installing/removing of necessary additional packages:

- Right click on the file -> Permissions -> Allow executing file as program:

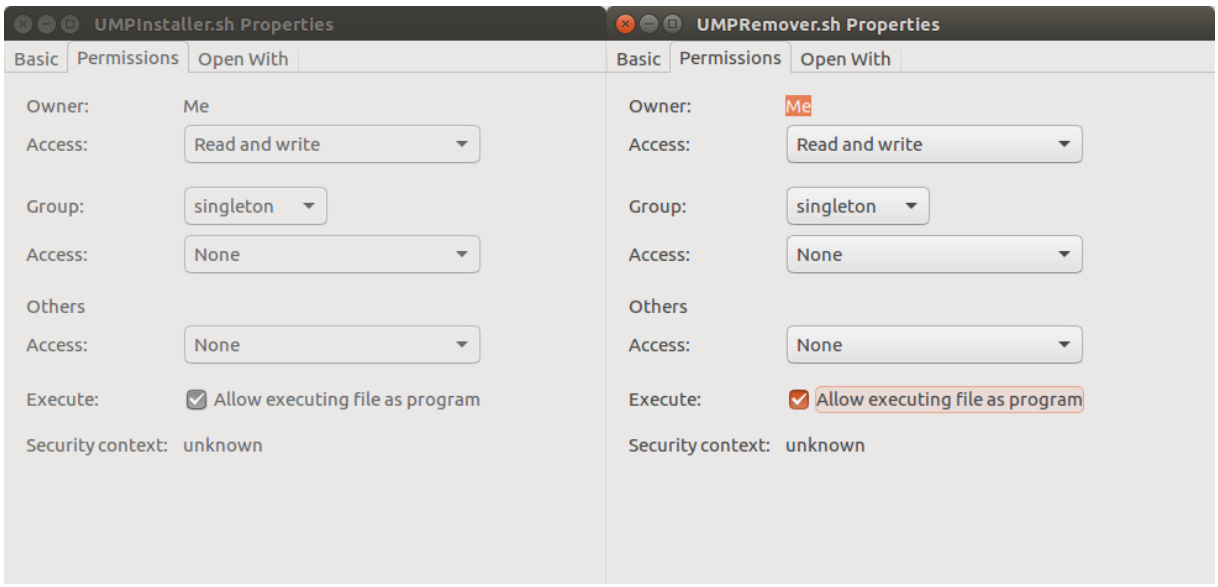


Figure 4.8: UMP setup special Linux shell scripts

Open terminal and go to the folder where you extract your build or make "Right click in build folder window" and click to the "Open in Terminal" option:

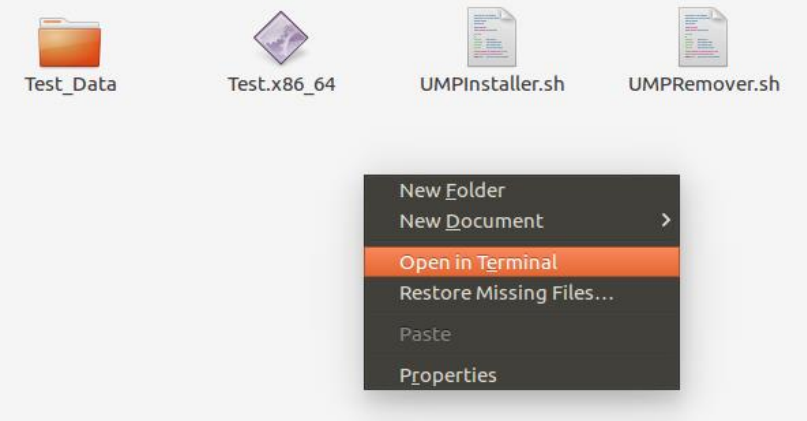


Figure 4.9: Open "Terminal" in build folder

In terminal you need to run special UMP shell script:

- To correct launch and install necessary packages: **./UMPInstaller.sh**
- To correct remove installed packages: **./UMPRemover.sh**

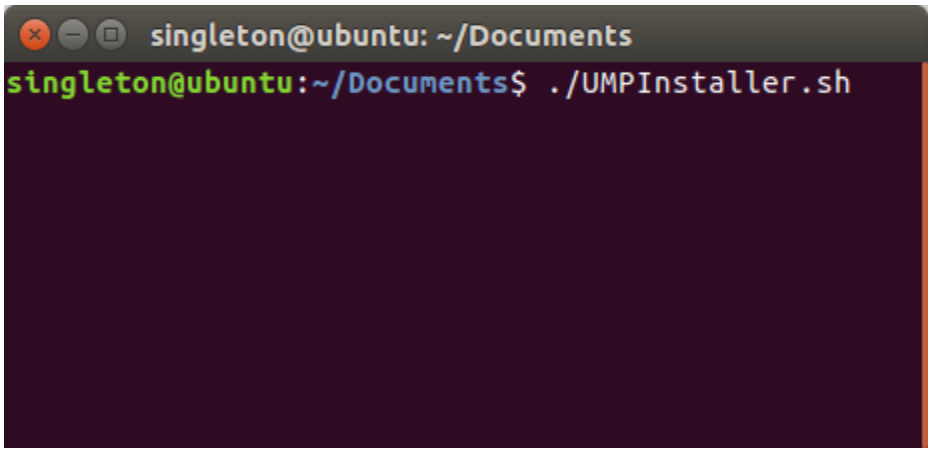


Figure 4.10: Command example in "Terminal"

If you have some problems to correctly run UMP shell scripts, please open it with default file editor and press "Save" button or "Ctrl+S" and try again. After first correct launch in future you can simply run your executable file without any additional manipulation.

IOS Platform Setup

When you completely export iOS project from Unity Engine, please check if all needed frameworks is available in "Build Phases" tab in "Link Binary With Libraries" phase:

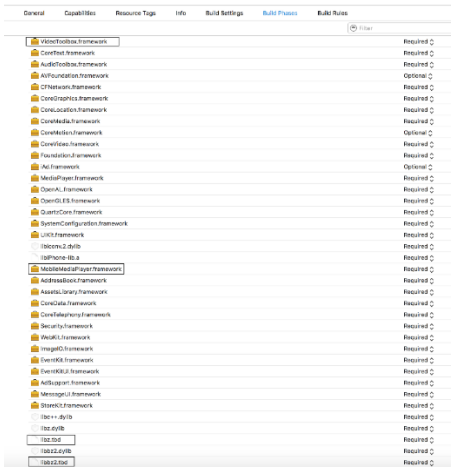


Figure 4.11: iOS build frameworks

If this frameworks and libraries (VideoToolbox, libz, libbz2) don't exist in your list, please add them manually or try to make Unity build.

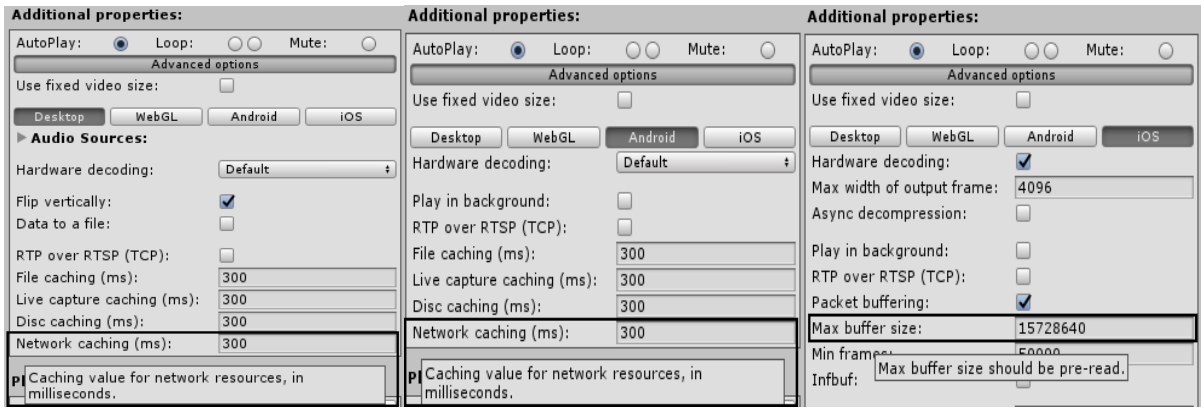
FAQ

What shader can I use?

You can use every shader that uses a texture property.

Can I change stream buffering time?

Just use “Additional properties” for platform where you will be check your build:



For Android platform you can also try to use this options (add this changes in "UniversalMediaPlayer.cs" script):

```
    case UMPSettings.Platforms.Android:
        var androidOptions = new PlayerOptionsAndroid(null)
        {
            FixedVideoSize = _useFixedSize ? new Vector2(_fixedVideoWidth,
_fixedVideoHeight) : Vector2.zero,
            HardwareDecoding = _hardwareDecodingAndroid,
            PlayInBackground = _playInBackgroundAndroid,
            UseTCP = _rtspOverTcpAndroid,
            FileCaching = _fileCachingAndroid,
            LiveCaching = _liveCachingAndroid,
            DiscCaching = _diskCachingAndroid,
            NetworkCaching = _networkCachingAndroid,
            ClockJitter = 0, //<- add this line
            ClockSynchro = 0 //<- add this line
        };

        options = androidOptions;
        break;
```

If this options don’t have effect, try to use different libVLC libraries (more information in Patches chapter).

Can't run Linux application correctly, what I am doing wrong?

If you see this message when you try to launch your build:

Could not display “appname”.

There is no application installed for “executable” files. Do you want to search for an application to open this file?

You need to make the file executable, just right click on the file -> Properties -> Permissions -> Allow executing file as program or from terminal: chmod +x appname

How to set up the audio output for Oculus Rift Headphones?

When you create new instance of media player object, just add additional defined argument to set new audio output device (add this changes in "UniversalMediaPlayer.cs" script):

```
    case UMPSettings.Platforms.Win:
    case UMPSettings.Platforms.Mac:
    case UMPSettings.Platforms.Linux:
        if (UMPSettings.EditorBitMode == UMPSettings.BitModes.x86 &&
            _hardwareDecodingDesktop == PlayerOptions.State.Default)
            _hardwareDecodingDesktop = PlayerOptions.State.Enable;

        var standaloneOptions = new PlayerOptionsStandalone(null)
        {
            FixedVideoSize = _useFixedSize ? new Vector2(_fixedVideoWidth,
_fixedVideoHeight) : Vector2.zero,
            AudioOutputSources = _audioSourcesDesktop,
            HardwareDecoding = _hardwareDecodingDesktop,
            FlipVertically = _flipVerticallyDesktop,
            UseTCP = _rtspOverTcpDesktop,
            FileCaching = _fileCachingDesktop,
            LiveCaching = _liveCachingDesktop,
            DiskCaching = _diskCachingDesktop,
            NetworkCaching = _networkCachingDesktop
        };

        if (_outputToFileDesktop)
            standaloneOptions.RedirectToFile(_displayOutputDesktop,
_outputFilePathDesktop);

        standaloneOptions.SetLogDetail(_logDetail, UnityConsoleLogging);
        standaloneOptions.SetAudioOutputDevice("Rift Audio"); //<- add this line
        options = standaloneOptions;
        break;
```

Why is my video playing in the editor but not on my mobile device?

Every platform has is own restrictions in terms of video codecs or audio formats. So it is most likely a problem of your video encoding.

How can I setup my custom GameObjecs with UMP (for example NGUI component):

You can use special callback to get video texture that will be created for current video and make some additional manipulation with it (how to use it you can find in “UniversalMediaPlayer.cs” script):

```
    public void OnPlayerPrepared(Texture2D videoTexture)
    {
        //apply video output texture to NGUI component
    }
```

Can I remove certain codecs such as .mpg (or any other that don't be used) from "Plugins" folder to decrease size on application?

Yes, you can remove some of them. For example you can run video that you want to support in your project and try to delete all "plugins" folder - so, all .dlls that used in current video playback you can't delete, but other .dlls will be deleted.

Is there any way to specify the resolution of the youtube videos?

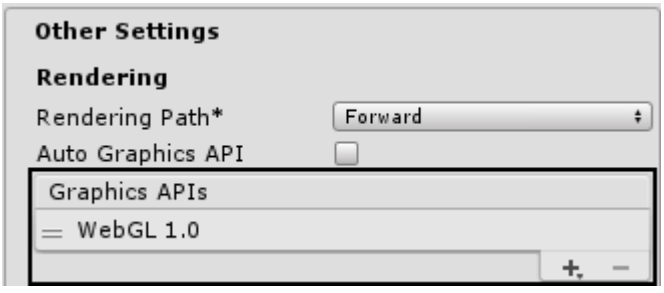
You have possibility to get video with resolution that you want, just use this function when you get youtube video link from youtube service:

```
var videoInfo = _videoHostingParser.GetBestQualityVideo(videoInfos, 240);//240, 360, 480, 720, 1080...
```

Can’t play video on WebGL platform (or how play streams on WebGL platform)?

As you know WebGL support is still experimental, and has problem thats named like: cross-domain policy problem (more information by this link: https://en.wikipedia.org/wiki/Cross-origin_resource_sharing) and appear when you try to play streams from other domain (on HTML + WebGL texture side), so if your server or domain is don't have **“Access-Control-Allow-Origin”** for your request it's will not work, what you can try is to write your own python (or other languages) server and used it to proxy requests for videos, which is not the most beautiful solution imaginable, but its effective. I'm hoping that a cleaner way to use texture sources from other domains is developed, because it's a really annoying problem. This problem is not actual if you will be playing the local video files or video files from **“StreamingAssets”** folder.

Also, please check your WebGL graphics api in **“Player Settings…”** and make the same like in screenshot below:



How do I setup 360 stereoscopic rendering (SBS or top-bottom)?

You can look how to do this in this tutorial: 360 Stereoscopic (top-bottom).

How get all possible additional player options and where I can get list of them?

All possible additional options that you can use with libVLC libraries you can find by this link: VLC command-line help (supported on desktop and Android platforms). Example of usage

```
    case UMPSettings.Platforms.Win:
    case UMPSettings.Platforms.Mac:
    case UMPSettings.Platforms.Linux:
        if (UMPSettings.EditorBitMode == UMPSettings.BitModes.x86 &&
            _hardwareDecodingDesktop == PlayerOptions.State.Default)
            _hardwareDecodingDesktop = PlayerOptions.State.Enable;

        var standaloneOptions = new PlayerOptionsStandalone(new string[] {
            "--http-proxy=http://proxy-host:proxy-port/",
            "--http-proxy-pwd=password" }) //<- string array with options
        {
            FixedVideoSize = _useFixedSize ? new Vector2(_fixedVideoWidth,
_fixedVideoHeight) : Vector2.zero,
            AudioOutputSources = _audioSourcesDesktop,
            HardwareDecoding = _hardwareDecodingDesktop,
            FlipVertically = _flipVerticallyDesktop,
            UseTCP = _rtspOverTcpDesktop,
            FileCaching = _fileCachingDesktop,
            LiveCaching = _liveCachingDesktop,
            DiskCaching = _diskCachingDesktop,
            NetworkCaching = _networkCachingDesktop
        };

        if (_outputToFileDesktop)
            standaloneOptions.RedirectToFile(_displayOutputDesktop,
_outputFilePathDesktop);

        standaloneOptions.SetLogDetail(_logDetail, UnityConsoleLogging);
        options = standaloneOptions;
        break;
```

How to load and use SRT files on desktop platforms?

You need to load your srt file (srt file should be in the same folder with video file), add this changes to "UniversalMediaPlayer.cs" script (if your SRT is builded into video file you can skip this step):

```
    case UMPSettings.Platforms.Win:
    case UMPSettings.Platforms.Mac:
    case UMPSettings.Platforms.Linux:
        if (UMPSettings.EditorBitMode == UMPSettings.BitModes.x86 &&
            _hardwareDecodingDesktop == PlayerOptions.State.Default)
            _hardwareDecodingDesktop = PlayerOptions.State.Enable;

        var standaloneOptions = new PlayerOptionsStandalone(new string[] { @"--sub-
file=C:\Users\Documents\UnityProjects\UniversalMediaPlayerTest\Assets\StreamingAssets\
sub.srt" })
        {
            FixedVideoSize = _useFixedSize ? new Vector2(_fixedVideoWidth,
_fixedVideoHeight) : Vector2.zero,
            AudioOutputSources = _audioSourcesDesktop,
            HardwareDecoding = _hardwareDecodingDesktop,
```

```
            FlipVertically = _flipVerticallyDesktop,
            UseTCP = _rtspOverTcpDesktop,
            FileCaching = _fileCachingDesktop,
            LiveCaching = _liveCachingDesktop,
            DiskCaching = _diskCachingDesktop,
            NetworkCaching = _networkCachingDesktop
        };

        if (_outputToFileDesktop)
            standaloneOptions.RedirectToFile(_displayOutputDesktop,
_outputFilePathDesktop);

        standaloneOptions.SetLogDetail(_logDetail, UnityConsoleLogging);
        options = standaloneOptions;
        break;
```

Choose the correct srt track, add this changes to "UniversalMediaPlayer.cs" script:

```
public void OnPlayerPrepared(Texture2D videoTexture)
{
    if (_mediaPlayer.SpuTracks.Length > 0)
    {
        _mediaPlayer.SpuTrack = _mediaPlayer.SpuTracks[_mediaPlayer.SpuTracks.Length -
1];
    }
    else
    {
        Debug.Log("Don't have any subtitle tracks");
    }
#if UNITY_EDITOR
    _lastEventMsg = "Prepared";
#endif
    if (_preparedEvent != null)
        _preparedEvent.Invoke(videoTexture);
}
```

CLASSES/METHODS DESCRIPTION

Classes

- `MediaPlayer` – media player class;
- `MediaPlayerHelper` – adds possibility to get some additional stuff from media player;
- `PlayerOptions` – used to setup new media player instance with some additional parameters (advanced usage).

Interfaces

- `IMediaListener` (combine all main video playback interfaces)
- `IPlayerOpeningListener` { `void OnPlayerOpening(); }`
- `IPlayerBufferingListener` { `void OnPlayerBuffering(float percentage); }`
- `IPlayerPreparedListener` { `void OnPathPrepared(string path, bool isPreparing); }`
- `IPlayerPlayingListener` { `void OnPlayerPlaying(); }`
- `IPlayerPausedListener` { `void OnPlayerPaused(); }`
- `IPlayerStoppedListener` { `void OnPlayerStopped(); }`
- `IPlayerEndReachedListener` { `void OnPlayerEndReached(); }`
- `IPlayerEncounteredErrorListener` { `void OnPlayerEncounteredError(); }`
- `IPlayerTimeChangedListener` { `void OnPlayerTimeChanged(long time); }`
- `IPlayerPositionChangedListener` { `void OnPlayerPositionChanged(float position);}`
- `IPlayerSnapshotTakenListener` { `void OnPlayerSnapshotTaken(string path); }`
- `ILogMessageListener` { `void OnLogMassage(PlayerManagerLogs.PlayerLog message); }`

MediaPlayer

Constructors

```
/// <summary>
/// Create new instance of media player object
/// </summary>
/// <param name="monoObject">MonoBehaviour instanse</param>
/// <param name="videoOutputObjects">Objects that will be rendering video
output</param>
public MediaPlayer(MonoBehaviour monoObject, GameObject[] videoOutputObjects)
```

```
/// <summary>
/// Create instance of MediaPlayer object with additional arguments
/// </summary>
/// <param name="monoObject">MonoBehaviour instanse</param>
/// <param name="videoOutputObjects">Objects that will be rendering video
output</param>
```

```
/// <param name="options">Additional player options</param>
public    MediaPlayer(MonoBehaviour    monoObject,    GameObject[]    videoOutputObjects,
PlayerOptions options)
```

Properties

```
/// <summary>
/// Get media player object for current running platform
/// (supported: Standalone, WebGL, Android and iOS platforms)
/// for get more additional possibilities that exists only for this platform.
/// </summary>
public object PlatformPlayer
```

```
/// <summary>
/// Get/Set simple array that consist with Unity 'GameObject' that have 'Mesh
Renderer' (with some material)
/// or 'Raw Image' component
/// </summary>
public GameObject[] VideoOutputObjects
```

```
/// <summary>
/// Get event manager for current media player to add possibility to attach/detach
special playback listeners
/// </summary>
public PlayerManagerEvents EventManager
```

```
/// <summary>
/// Get player additional options for current running platform
/// </summary>
public PlayerOptions Options
```

```
/// <summary>
/// Get current video playback state
/// </summary>
public PlayerState State
```

```
/// <summary>
/// Get current video playback state value (can be float, long or string type)
/// </summary>
public object StateValue
```

```
/// <summary>
/// Get/Set local path or url link to your video/audio file/stream
/// Example of using:
/// Local storage space - 'file:///C:\MyFolder\Videos\video1.mp4' or
/// 'C:\MyFolder\Videos\video1.mp4' or
/// 'file:///DCIM/100ANDRO/MyVideo.mp4' (example for Android platform);
/// Remote space (streams) -
'rtsp://wowzaec2demo.streamlock.net/vod/mp4:BigBuckBunny_115k.mov';
/// 'StreamingAssets' folder - 'file:///myVideoFile.mp4';
/// </summary>
public Uri DataSource
```

```
/// <summary>
```

```
/// Is media is currently playing
/// </summary>
public bool IsPlaying
-----------------------------------------------------------------------------------------------------

/// <summary>
/// Is media is ready to play (first frame available)
/// </summary>
public bool IsReady
-----------------------------------------------------------------------------------------------------

/// <summary>
/// Is the player able to play
/// </summary>
public bool AbleToPlay
-----------------------------------------------------------------------------------------------------

/// <summary>
/// Get the current video length (in milliseconds)
/// </summary>
public long Length
-----------------------------------------------------------------------------------------------------

/// <summary>
/// Get frames per second (fps) for current video playback.
/// * Warning: it's not a predefined value from video file/stream - calculated in
video playback process
/// </summary>
public float FrameRate
-----------------------------------------------------------------------------------------------------

/// <summary>
/// Get video frames counter
/// </summary>
public int FramesCounter
-----------------------------------------------------------------------------------------------------

/// <summary>
/// Get pixels of current video frame
/// Example of using:
/// texture.LoadRawTextureData(_player.FramePixels);
/// texture.Apply();
/// </summary>
public byte[] FramePixels
-----------------------------------------------------------------------------------------------------

/// <summary>
/// Get/Set the current video time (in milliseconds).
/// This has no effect if no media is being played.
/// Not all formats and protocols support this
/// </summary>
public long Time
-----------------------------------------------------------------------------------------------------

/// <summary>
/// Get/Set video position.
/// This might not work depending on the underlying input format and protocol
/// </summary>
public float Position
-----------------------------------------------------------------------------------------------------
```

```
/// <summary>
/// Get/Set the requested video play rate
/// </summary>
public float PlaybackRate
```

```
/// <summary>
/// Get/Set current software audio volume
/// (by default you can change this value from '0' to '100')
/// </summary>
public int Volume
```

```
/// <summary>
/// Get/Set mute status for current video playback
/// </summary>
public bool Mute
```

```
/// <summary>
/// Get current video width in pixels
/// </summary>
public int VideoWidth
```

```
/// <summary>
/// Get current video height in pixels
/// </summary>
public int VideoHeight
```

```
/// <summary>
/// Get the pixel dimensions of current video
/// </summary>
public Vector2 VideoSize
```

```
/// <summary>
/// Get/Set the current audio track
/// </summary>
public MediaTrackInfo AudioTrack
```

```
/// <summary>
/// Get the available audio tracks
/// </summary>
public MediaTrackInfo[] AudioTracks
```

```
/// <summary>
/// Get/Set the current spu (subtitle) track (supported only on Standalone platform)
/// </summary>
public MediaTrackInfo SpuTrack
```

Methods

```
/// <summary>
/// Add new main group of listeners to current media player instance
/// </summary>
/// <param name="listener">Group of listeners</param>
```

```
public void AddMediaListener(IMediaListener listener)
-----------------------------------------------------------------------------------------------------

/// <summary>
/// Remove group of listeners from current media player instance
/// </summary>
/// <param name="listener">Group of listeners</param>
public void RemoveMediaListener(IMediaListener listener)
-----------------------------------------------------------------------------------------------------

/// <summary>
/// Prepare current video playback
/// </summary>
public void Prepare()
-----------------------------------------------------------------------------------------------------

/// <summary>
/// Play or resume (True - playback started (and was already started), False on error.
/// </summary>
/// <returns></returns>
public bool Play()
-----------------------------------------------------------------------------------------------------

/// <summary>
/// Toggle pause current video playback (no effect if there is no media)
/// </summary>
public void Pause()
-----------------------------------------------------------------------------------------------------

/// <summary>
/// Stop current video playback (no effect if there is no media)
/// </summary>
/// <param name="resetTexture">Clear the last frame</param>
public void Stop(bool resetTexture)
-----------------------------------------------------------------------------------------------------

/// <summary>
/// Stop current video playback (no effect if there is no media)
/// </summary>
public void Stop()
-----------------------------------------------------------------------------------------------------

/// <summary>
/// Release a current media player instance
/// </summary>
public void Release()
-----------------------------------------------------------------------------------------------------

/// <summary>
/// Get the current video formatted length (hh:mm:ss[:ms]).
/// </summary>
/// <param name="detail">True: formatted length will be with [:ms]</param>
public string GetFormattedLength(bool detail)
-----------------------------------------------------------------------------------------------------

/// <summary>
/// Set new video subtitle file
/// </summary>
/// <param name="path">Path to the new video subtitle file</param>
/// <returns></returns>
public bool SetSubtitleFile(Uri path)
```

MediaPlayerHelper

```
/// <summary>
/// Apply texture to Unity game objects that has 'RawImage' or 'MeshRenderer'
component
/// </summary>
/// <param name="texture">Texture to render video output</param>
/// <param name="renderingObjects">Game objects where will be rendering video
output</param>
/// <returns></returns>
public static void ApplyTextureToRenderingObjects(Texture2D texture, GameObject[]
renderingObjects)
```

```
/// <summary>
/// Generate correct texture for current runtime platform
/// </summary>
/// <param name="width">Width in pixels</param>
/// <param name="height">Height in pixels</param>
/// <returns></returns>
public static Texture2D GenVideoTexture(int width, int height)
```

```
/// <summary>
/// Getting average color from frame buffer array
/// </summary>
/// <returns></returns>
public static Color GetAverageColor(byte[] frameBuffer)
```

```
/// <summary>
/// Getting colors from frame buffer array
/// </summary>
/// <returns></returns>
public static Color32[] GetFrameColors(byte[] frameBuffer)
```

```
/// <summary>
/// Getting root storage menory for current platform
/// Windows, Mac OS, Linux will return path to project folder
/// Android, iOS will return root of internal/external memory root
/// </summary>
/// <returns></returns>
public static string GetDeviceRootPath()
```

```
/// <summary>
/// Check if file exists in 'StreamingAssets' folder
/// </summary>
/// <param name="fileName">File name or path to file in 'StreamingAssets'
folder</param>
/// <returns></returns>
public static bool IsAssetsFile(string filePath)
```