UMP
User Manual

Version 1.3

INTRODUCTION

UMP (Universal Media Player) is cross platform universal media framework for Windows, Linux, OSX, Android, iOS 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)
Linux	x86/x86_64/Universal	VLC	Ubuntu 12.04.5 LTS+ (OpenGL)
OSX	x86_x64	VLC	10.10 Yosemite+ (OpenGL)
Android	armeabi-v7a/x86	VLC	Android API level 14+ (Android 4.0+) (OpenGLES 2.0 or 3.0)
iOS	arm64, armv7	FFmpeg	iOS 6+ (OpenGLES 2.0 or Metal)

Fully compatible with Unity Editor in different modes (fast native texture updates).

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.

Supported main video player events: Opening, Buffering, Playing, Paused, Stopped, Ended, Error.

Supported full logging system from native library in Unity Editor for more debugging possibility with different depth: Warning, Debug.

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

1. Import the UMP package from the AssetStore. You should now have a folder named UMP with the following structure in your Unity project:

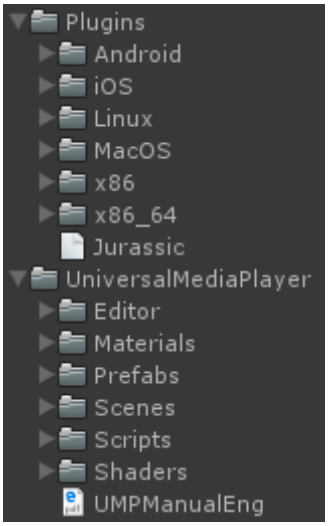

Figure 2.1: package structure

- Plugins: All the native dlls and shared libraries;
- Editor: Custom Editor/build script;
- Materials: Custom materials that used in example scenes that based on custom shaders;
- Prefabs: Special UMP prefab that used for easy setup your project with UMP asset;
- 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.

2. When updating an existing UMP installation you should delete the old UMP and Plugins folder (don't forget to store your custom plugins) before importing. 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.
3. UMP asset consists of two parts: UMP (Win, Mac, Linux) - Standalone and UMP (Android, iOS) - Mobile, so if you bought only mobile version it's will not work in Unity Editor mode and you will see this message in Unity Console:
    - Don't support video playback in Unity Editor without UMP (Win, Mac, Linux) package!!!

   So if you wanna editor support you need to buy also UMP Standalone.

USAGE

1. 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:

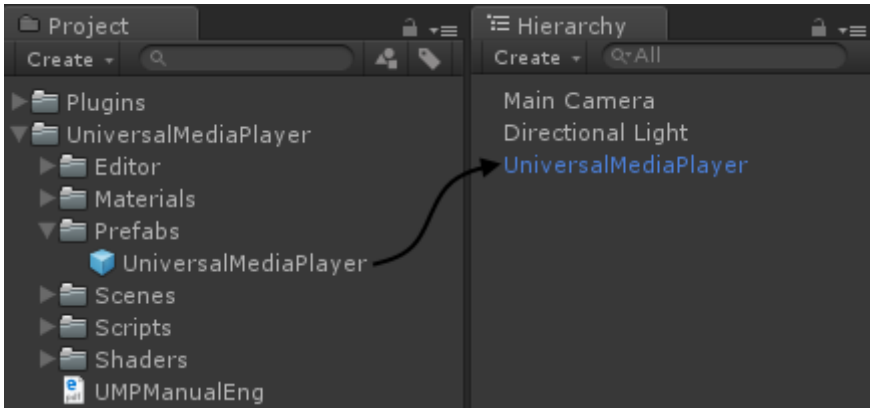

Figure 3.1: add new UMP instance

2. When you select the UMP instance in your hierarchy window you can manage it with the component inspector:

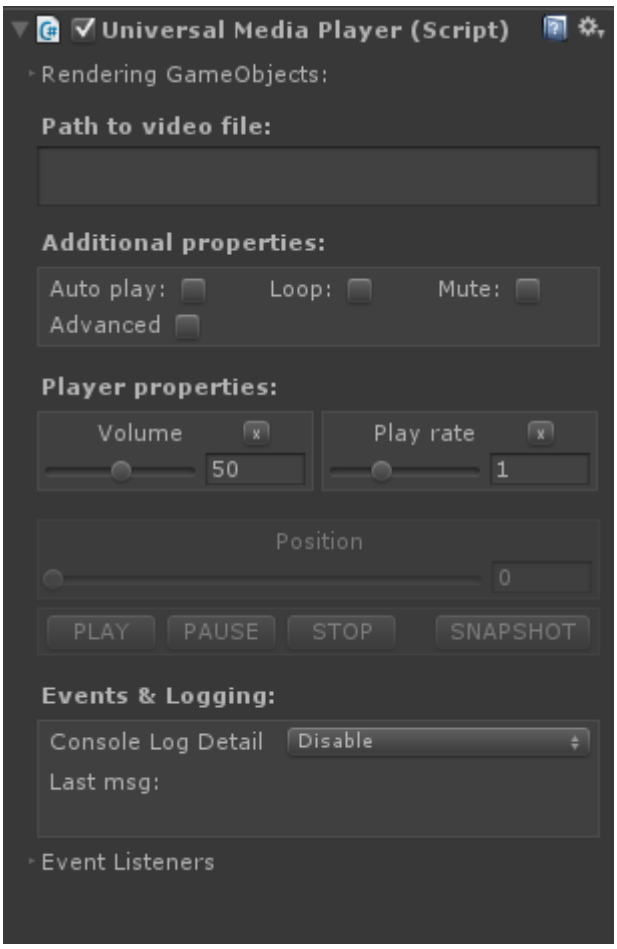

Figure 3.2: UMP inspector view

3. Rendering GameObjects: Simple array that consist with Unity “GameObjects” that have "Mesh Renderer" with some material or "Raw Image" component:

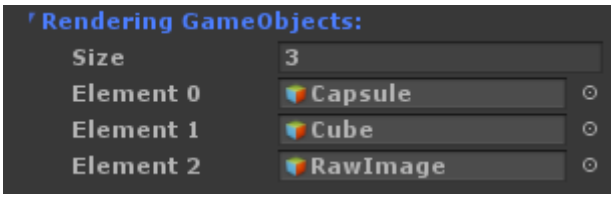

Figure 3.3: UMP rendering objects

4. Path to video file: url link or local path to your video. Both of this “path” will be work correctly and you don't need to worry about video file location. You can change the videoPath and press a "Play" button at every time. 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.

   Examples:

   - file:///myVideoFile.mp4 -> StreamingAssets\myVideoFile.mp4;
   - file:///C:\MyFolder\Videos\video1.mp4 -> C:\MyFolder\Videos\video1.mp4;
   - C:\MyFolder\Videos\video1.mp4 -> C:\MyFolder\Videos\video1.mp4;
   - rtsp://wowzaec2demo.streamlock.net/vod/mp4:BigBuckBunny_115k.mov

   If you want to load video file on **Android platform** from an external sdcard you have to use a path like this (for "StreamingAssets" folder you can use previous example):

   - file:///DCIM/100ANDRO/MyVideo.mp4 -> /storage/emulated/0/DCIM/100ANDRO/MyVideo.mp4

   So you don't need to use in your path root name of device sd card, because on some Android devices it can be different, also don't forget to set 'Write Access' to 'External (SDCard) in your player settings.

   If you want to load video file on **iOS platform** from an external memory you need to download the special "MobileMediaPlayer.framework", that contains all possible supported codecs (also file reading protocols) and replace to old one that located in "Plugins\iOS" folder. This updated framework not contains by default, because he has very big size for "Assets Store" and you should download it separately.

   Link to download: MobileMediaPlayer.framework

5. Additional properties:
   - **Autoplay:** Start playback automatically after video is buffered;
   - **Looping:** When the playback reaches the end position it jumps to the start and plays again;
   - **Mute:** Set mute status for current video playback;
   - **Advanced:** Special properties that give you possibility to have additional setup for your video playback (file, live capture, disc and network caching).
6. Player properties:
   - **Play:** Start video playback;

- **Pause:** If a video is playing you can pause it. To continue playback you have to press "Play" button again;
- **Stop:** Stops the media and seeks to the first frame. You need to press "Play" button to start playback again;
- **Snapshot:** Supported only on "Standalone" platforms (Win, Mac, Linux). Take a snapshot of the current video playback and save it in Application.persistentDataPath folder.
- **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;
- **Volume:** Set current software audio volume;
- **Play rate:** Set the requested movie play rate (don't support negative playback).

7. Events & Logging:
    - **Console Log Detail:** Supported only on "Standalone" platforms. Gives you possibility to getting log messages from native library, possible filters: Debug, Warning, Error;
    - **Last msg:** Display last playback event or playback error;
    - **Event listeners:** Gives possibility to attach to the playback callbacks (Opening, Buffering, Playing, Paused, Stopped, End Reached, Encountered Error, Time Changed, Position Changed, Snapshot events).

BUILD INSTRUCTIONS

Linux:

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

So first of all please add "Execute" permission to special UMP shell scripts "UMPLauncher.sh" and "UMPRemover.sh" - this scripts should be generated automatically when Unity build process will be correctly finished. They give you possibility to automate installing/removing of necessary packages:

Right click on the file -> go to "Permissions" -> check "Allow executing file as program"

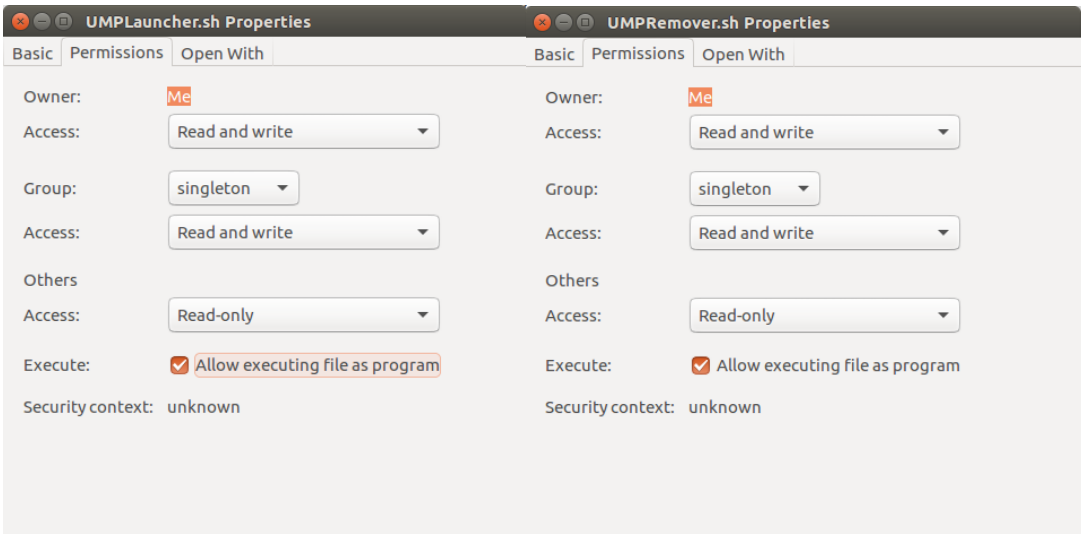

Figure 4.1: UMP setup special Linux shell scripts

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

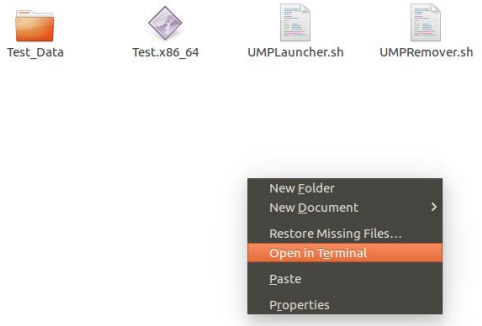

Figure 4.2: Open "Terminal" in build folder

In terminal you need to run UMP shell script:

- To correct launch and install necessary packages: './UMPLauncher.sh'

- To correct remove installed packages: './UMPRemover.sh'

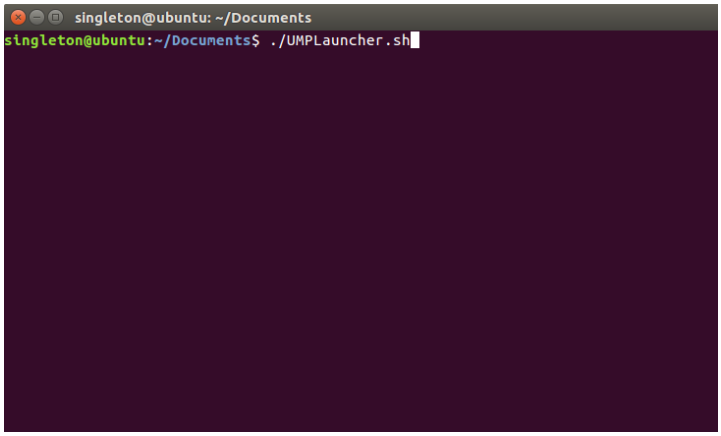

Figure 4.3: Command example in "Terminal"

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 running shell scripts.

iOS:

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

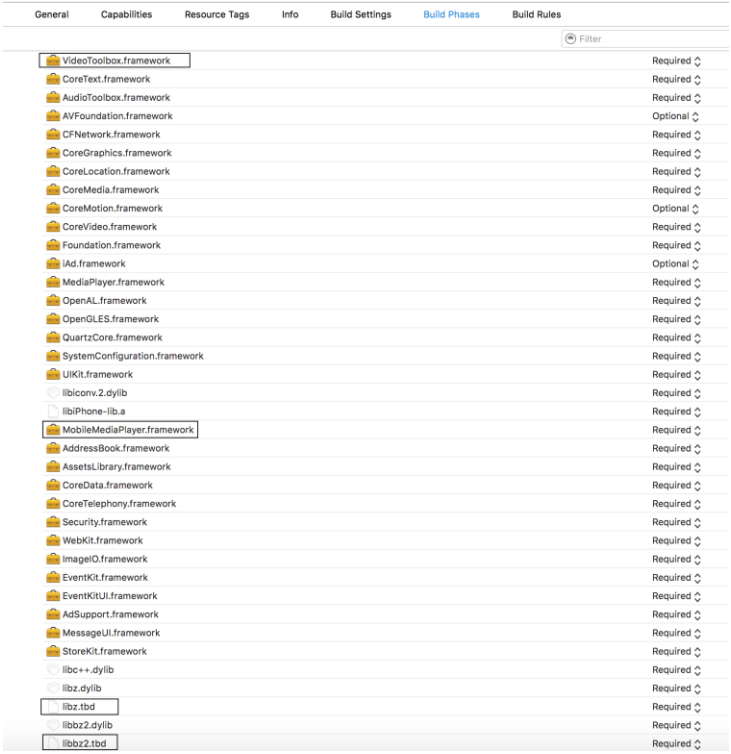

Figure 4.4: iOS build frameworks

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

FAQ

- *What shader can I use?*

  You can use every shader that uses a texture property.

- *Don't see any video output only sound on Android platform, what I am doing wrong?*

  Please uncheck the "Multithreaded Rendering" property in "Player Settings" - it's don't support in current version.

- *Can I change stream buffering size for iOS platform*

  Yes you can do this by editing "MobileMediaPlayer.mm" file that exist in "Libraries/Plugins/iOS/" and add new option "max-buffer-size" when create new media player object in "initMediaPlayer" method:

  ```
  #define MAX_QUEUE_SIZE (15 * 1024 * 1024)

  //...

  [options setPlayerOptionIntValue:1 forKey:@"start-on-prepared"];

  [options setPlayerOptionIntValue:1 forKey:@"videotoolbox"];

  [options setPlayerOptionIntValue:MAX_QUEUE_SIZE forKey:@"max-buffer-size"];

  _mediaPlayer = [[FFMoviePlayerController alloc] initWithOptions:options];
  ```

  All possible options you can find in this file: Additional Options

- *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** -> check **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:

  ```
  AudioOutputDevice = DefinedArgs.GetAudioOutputDevice("Rift Audio")
  ```

- *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:

Callback example:
public void OnPlayerTextureCreated(Texture2D videoTexture)
{

```
//apply video output texture to NGUI component
```

}

- *Can I play movies on a texture that I create?*

  Texture that will be used to show video output is created automatically (because different video has different resolution and many devices can't store many textures in memory, so it's will be automatically destroy if it not used anymore) and handled all compared operation with it, so you don't need to make it directly.

  Constructor example:
  MediaPlayer(MonoBehaviour monoObject, GameObject[] outputObjects)
  Create and initialize a MediaPlayer instance:

  - monoObject – set MonoBehaviour instance, script where MediaPlayer will be using
  - outputObjects – objects that will be rendering video output

- *I was unable to play any videos from the editor, do I need some additional codec package on my system, or something else?*

  If you have only UMP Mobile version (UMP (Android, iOS) without UMP Standalone version (UMP (Win, Mac, Linux) it's don't support Unity Editor video playback, also you don't need any additional codecs, all main codecs is available when you import package to your project.

- *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 wanna support 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:

  Example of getting youtube video with best resolution:

  ParsedVideoInfo videoWithBestResolution = new ParsedVideoInfo(0);

  foreach (var info in videoInfos)

  {

  ```
  if (info.Resolution > videoWithBestResolution.Resolution)
  ```

  ```
  videoWithBestResolution = info;
  ```

  }

KNOWN ISSUES

- There are some platform differences in terms of how remote files could be played or how they are cached on the system. Please check the video behaviour on all platforms you use. Some features don't work exactly the same way in the editor like on mobile platforms.
- On Android platform there is an issue with "Multithreaded Rendering" support, so please uncheck it before make your build.
- Don't support "Youtube Live Streaming", also can't play some youtube video that has cipher signature, because it's little complicated to get direct link to video on mobile platforms (Android, iOS), but it's will be fixed in near future.
- Some features is supported on one platform but don't support on another.

SUPPORT

If you need support or have any question/suggestions please contact us.

• Emails:

Gmail

Additional(Outlook)

CLASSES/METHODS DESCRIPTION

Classes:

- StandaloneMediaPlayer – media player for standalone platforms
- MobileMediaPlayer – media player for mobile platforms
- MediaPlayer – combines standalone and mobile media players
- MediaPlayerHelper – adds possibility to get some additional stuff from media player
- DefinedArgs – 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)
- IPlayerTextureCreatedListener
    - void OnPlayerTextureCreated(UnityEngine.Texture2D videoTexture)
- 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(LogMessage message)

MediaPlayer

- void AddMediaListeners(IMediaListeners listener)
  Add to MediaPlayer new main group of listeners.

- void RemoveMediaListener(IMediaListener listener)
  Remove from MediaPlayer the main group of listeners.

- void SetDataSource(Uri path)
  Set path to video source that will be use do play:
    - *path – path to video source*

- bool SetSubtitleFile(Uri path)
  Set new video subtitle file
    - *path – path to new video subtitle file*

- GameObject[] VideoOutputObjects { get; set; }
  Objects that will be rendering video output

- MediaEventManager EventManager { get; }
  Adds possibility to attach/detach of media player listeners

- int GetVideoWidth()
  Get current video width in pixels

- int GetVideoHeight()
  Get current video height in pixels

- Vector2 GetVideoSize()
  Get the pixel dimensions of a video:

- *new Vector2(the pixel width, the pixel height)*

- byte[] FramePixels { get; }
  Get pixels of current frame

- float Fps { get; }
  Get movie fps rate:
  - *frames per second (fps) for this playing movie*

- int FrameCounter { get; }
  Get movie frame counter

- bool Play()
  Play or resume:
  - *True if playback started (and was already started), or False on error*

- void Pause()
  Toggle pause (no effect if there is no media)

- void Stop()
  Stop (no effect if there is no media)

- void Stop(bool clearVideoTexture)
  Stop (no effect if there is no media)
  - *clearVideoTexture – clear the last frame of current movie. Used for looping video playback*

- void Release()
  Release a MediaPlayer instance

- long GetLength()
  Get the current movie length (in ms)

- string GetFormattedLength(bool detail)
  Get the current movie formatted length (hh:mm:ss[:ms]):
  - *detail – is formatted length will be with [:ms]*

- TrackDescription[] SpuTracks { get; }
  Gets the available spu tracks

- TrackDescription SpuTrack { get; set; }
  Gets/Sets the current spu track

- TrackDescription[] AudioTracks { get; }
  Gets the available audio tracks

- TrackDescription AudioTrack { get; set; }
  Gets/Set the current audio track

- int Volume { get; set; }
  Get/Set current software audio volume:
  - *the software volume in percents (0 = mute, 100 = nominal / 0dB)*

- long Time { get; set; }
  Get/Set the current movie time (in ms). This has no effect if no media is being played. Not all formats and protocols support this:
  - *the movie time (in ms), or -1 if there is no media*

- float Position { get; set; }
  Get/Set movie position. This has no effect if playback is not enabled. This might not work depending on the underlying input format and protocol:
  - *movie position, or -1. in case of error*

- bool VideoTextureExist { get; }
  Is video texture exist

- bool Mute { get; set; }
  Get/Set current mute status:
  - *the mute status*

- bool IsPlaying { get; }
  Is media is currently playing:
  - *True if the player is playing, False otherwise*

- bool IsReady { get; }
  Is media is ready to play (first frame available):
  - *True if the player is playing, False otherwise*

- bool AbleToPlay { get; }
  Is the player able to play:
  - *True if the player is able to play, False otherwise*

- float PlaybackRate { get; set; }
  Get/Set the requested movie play rate:
  - *-1 if an error was detected, 0 otherwise (but even then, it might not actually work depending on the underlying media protocol)*

StandaloneMediaPlayer

Based on MediaPlayer class.

- StandaloneMediaPlayer(MonoBehaviour monoObject, GameObject[] videoOutputObjects)
  Create and initialize a MediaPlayer instance:
  - *monoObject – set MonoBehaviour instance, script where MediaPlayer will be using*
  - *videoOutputObjects – objects that will be rendering video output*

- StandaloneMediaPlayer(MonoBehaviour monoObject, GameObject[] videoOutputObjects, DefinedArgs args)
  Create and initialize a MediaPlayer instance with special arguments:
  - *monoObject – set MonoBehaviour instance, script where MediaPlayer will be using;*
  - *videoOutputObjects – objects that will be rendering video output*

- *args - special defined arguments (more information about arguments you can find by clicking the link)*

- StandaloneMediaPlayer(MonoBehaviour monoObject, GameObject[] videoOutputObjects, string[] args)
  Create new instance of MediaPlayer object with special arguments:
  - *monoObject – MonoBehaviour instance*
  - *videoOutputObjects – objects that will be rendering video output*
  - *args - special arguments to additional setup*

- StandaloneMediaPlayer(MonoBehaviour monoObject, int fixedWidth, int fixedHeight, GameObject[] videoOutputObjects)
  Create new instance of MediaPlayer object with fixed video size:
  - *monoObject – MonoBehaviour instance*
  - *fixedWidth - fixed video width - rescale video width*
  - *fixedHeight - fixed video height - rescale video height*
  - *videoOutputObjects – objects that will be rendering video output*

- StandaloneMediaPlayer(MonoBehaviour monoObject, int fixedWidth, int fixedHeight, GameObject[] videoOutputObjects, DefinedArgs args)
  Create new instance of MediaPlayer object with fixed video size and special arguments:
  - *monoObject – MonoBehaviour instance*
  - *fixedWidth - fixed video width - rescale video width*
  - *fixedHeight - fixed video height - rescale video height*
  - *videoOutputObjects – objects that will be rendering video output*
  - *args - special defined arguments (more information about arguments you can find by clicking the link)*

- StandaloneMediaPlayer(MonoBehaviour monoObject, int fixedWidth, int fixedHeight, GameObject[] videoOutputObjects, string[] args)
  Create new instance of MediaPlayer object with fixed video size and special arguments:
  - *monoObject – MonoBehaviour instance*
  - *fixedWidth - fixed video width - rescale video width*
  - *fixedHeight - fixed video height - rescale video height*
  - *videoOutputObjects – objects that will be rendering video output*
  - *args - special defined arguments (more information about arguments you can find by clicking the link)*

- float VideoScale { get; set; }
  Get/Set the current video scaling factor. That is the ratio of the number of pixels on screen to the number of pixels in the original decoded video in each dimension. Zero is a special value; it will adjust the video to the output window/drawable (in windowed mode) or the entire screen. Note that not all video outputs support scaling:
    - *the currently configured zoom factor, or 0 if the video is set to fit to the output window/drawable automatically*

- string Version { get; }
  Retrieve libvlc version. Example: "1.1.0-git The Luggage"

- bool FlipVertically { get; set; }
  Use programmatically flip frame when we get it from native library

- void TakeSnapShot(string path)
  Take a snapshot of the current video window:
    - *path - the path where to save the screenshot to*

- string GetLastError()
  Get last logged error:
    - *error message*

MobileMediaPlayer

Based on MediaPlayer class and has the same constructors like in StandaloneMediaPlayer class.

MediaPlayerHelper

- static void ApplyTextureToRenderingObjects(Texture2D texture, GameObject[] renderingObjects)
  Apply texture to Unity game objects that has 'RawImage' or 'MeshRenderer' component
    - *texture - texture video output*
    - *renderingObjects - Game objects where will be rendering video output*

- static Color GetAverageColor(byte[] frameBuffer)
  Getting average color from frame buffer array (FramePixels)