RENDERHEADS

AVPro Video

for Android, iOS, tvOS, macOS, WebGL
Windows Desktop, Windows Phone and UWP
Unity plugin for fast and flexible video playback

USER MANUAL

Version 1.11.2

Released 28th May 2020

Contents

1. Introduction

AVPro Video is the newest video playback plugin from RenderHeads. We previously developed the AVPro QuickTime and AVPro Windows Media plugins for Unity. In this next generation of plugins we aim to create an easy to use, cross-platform video playback system that uses the native features of each platform.

1.1 Features

- Versions for iOS, tvOS, macOS, Android, WebGL, Windows, Windows Phone and UWP
- One API for video playback on all supported platforms
- *NEW* Facebook Audio 360 and Android ExoPlayer support
- 8K video support (on supported hardware)
- VR support (mono, stereo, equirectangular and cubemap)
- Transparency support (native and packed)
- Unity Pro 4.6.x and above supported
- Unity Personal 5.x - 2019.1.x and above supported
- Free watermarked trial version available (download here)
- Fast native Direct3D, OpenGL and Metal texture updates
- Linear and Gamma colour spaces supported
- Graceful fallback in editor
- Components for IMGUI, uGUI and NGUI
- Extensive PlayMaker support
- Easy to use, drag and drop components
- Desktop support for Hap, Hap Alpha, Hap Q and Hap Q Alpha
- Streaming and adaptive video from URL

1.2 Trial Version

We offer an unlimited trial version of AVPro Video for download from our website at http://renderheads.com/product/avpro-video/. The trial version has no missing features or time restrictions but it does apply a watermark to the rendered output. The watermarking does have a small performance impact which is only really noticeable on very high resolution videos.

1.3 Media Credits

BigBuckBunny_360p30.mp4 - (c) copyright 2008, Blender Foundation / www.bigbuckbunny.org

BigBuckBunny_720p30.mp4 - (c) copyright 2008, Blender Foundation / www.bigbuckbunny.org

SampleSphere.mp4 - (c) copyright Daniel Arnett, https://vimeo.com/97887646

2. System Requirements

- Unity
    - 2019.x, 2018.x, 2017.x, 5.6 Personal Edition - all supported platforms
    - 4.6 Pro - all supported platform except iOS, tvOS and macOS
- Platforms
    - iOS 8.1 and above
    - tvOS 9.0 and above
    - macOS 10.8 and above, 64-bit only
    - Android 4.0.3 (Ice Cream Sandwich, API level 15) and above (ARM7, ARM8 and x86)
    - Windows XP (SP 3) and above (32-bit and 64-bit)
    - Windows 8.0 and above (32-bit and 64-bit)
    - Windows Phone UWP 8.1 (32-bit and ARM)
    - Windows Desktop UWP 8.1 (32-bit, 64-bit and ARM)
    - Universal Windows Platform 10 (32-bit, 64-bit, ARM and ARM64)
    - WebGL compatible browser (*see notes for Safari issue)

2.1 VR / AR / MR / XR Headsets Supported

- Microsoft Hololens & Hololens 2 (UWP)
- Windows mixed-reality headsets (UWP):
    - Samsung Odyssey
    - Asus
    - HP
    - Acer
    - Lenovo Explorer
    - Dell Visor
- HTC Vive (Windows desktop)
- HTC Vive Pro (Windows desktop)
- HTC Vive Focus (Android)
- HTC Vive Cosmos (Windows desktop)
- Valve Index (Windows desktop)
- Oculus Rift (Windows desktop)
- Oculus Rift Go (Android)
- Oculus Rift S (Windows desktop)
- Oculus Gear VR (Android)
- Oculus Quest (Android)
- Google Cardboard (Android)
- Google Daydream (Android)
- Pico Goblin & Neo (Android)
- StarVR (Windows desktop)
- Lenovo Mirage Solo (Android)

2.2 Platforms not Supported

- WebPlayer
- Linux desktop
- Tizen
- Samsung TV
- Game Consoles (XBox*, PS4 etc)

** XBox One may be supported using UWP build option. We have not tested this though.*

3. Installation

1. Open up a fresh Unity session (to clear any locked plugin files)
2. Import the **unitypackage** file into your Unity project. If prompted to upgrade some scripts click Yes.

3.1 Trial Version & Watermark Notes

3.1.1 Watermark

If you are using a trial version of the plugin then you will see a watermark displayed over the video. The watermark is in the form of a “RenderHeads” logo that animates around the screen, or a thick horizontal bar that moves around the screen.

The full version of AVPro Video has no watermarks for any platforms. If you use one of the platform specific packages (eg AVPro Video for iOS, or AVPro Video for Windows) then you will not see the watermark on the platform you purchased for, but you will see the watermark on the other platforms. For example if you purchased AVPro Video for iOS then you will still see the watermark in the Unity editor as this is running on Windows/macOS, but the videos played back when you deploy to your iOS device will be watermark-free.

3.1.2 Updating from Trial Version

If you are upgrading from the trial version, make sure you delete the old /Assets/Plugins folder as this contains the trial plugin and could conflict. You may need to close Unity first, delete the files manually and then restart Unity and re-import the package (because Unity locks native plugin files once they are loaded).

You can check which version you have installed by adding an MediaPlayer component to your scene and clicking on the ‘about’ button in the Inspector for that component. The version number is displayed in this box.

3.2 Installing Multiple Platform Packages

If you are not using the full all-in-one AVPro Video package and instead have opted to

purchase multiple individual platform packages then the installation must be done carefully, especially when upgrading to a new version.

If you have installed the iOS package then it will also contain plugins for all of the other platforms but with the watermark enabled. This means that if you then try to install another AVPro Video package it may not override the plugins correctly. Here is how to resolve this using the iOS and Android package as examples:

1. Open a fresh Unity instance (this is important as otherwise Unity may have locked the plugin files which prevents them from being upgraded)
2. Import the iOS package
3. Import the Android package, but make sure that you have the iOS native plugin file unticked (so that it is not overwritten)

   A similar process can be applied for other package combinations.

List of native plugin files:

- Android
  - Plugins/Android/AVProVideo.jar
  - Plugins/Android/libs/armeabi-v7a/libAVProLocal.so
  - Plugins/Android/libs/arm64-v8a/libAVProLocal.so
  - Plugins/Android/libs/x86/libAVProLocal.so
- macOS
  - Plugins/AVProVideo.bundle
- iOS
  - Plugins/iOS/libAVProVideoiOS.a
- tvOS
  - Plugins/tvOS/libAVProVideotvOS.a
- WebGL
  - Plugins/WebGL/AVProVideo.jslib
- Windows
  - Plugins/WSA/PhoneSDK81/ARM/AVProVideo.dll
  - Plugins/WSA/PhoneSDK81/x86/AVProVideo.dll
  - Plugins/WSA/SDK81/ARM/AVProVideo.dll
  - Plugins/WSA/SDK81/x86/AVProVideo.dll
  - Plugins/WSA/SDK81/x86_64/AVProVideo.dll
  - Plugins/WSA/UWP/ARM/AVProVideo.dll
  - Plugins/WSA/UWP/x86/AVProVideo.dll
  - Plugins/WSA/UWP/x86_64/AVProVideo.dll
  - Plugins/x86/AVProVideo.dll
  - Plugins/x86_64/AVProVideo.dll

4. Usage Notes

4.1 Platform Notes

Most graphics APIs are supported:

	D3D9	D3D11	OpenGL	GLES2.0	GLES3.0	Metal	Vulkan
Android	N/A	N/A	N/A	**Yes**	**Yes**	N/A	No
iOS / tvOS	N/A	N/A	N/A	**Yes**	**Yes**	**Yes**	No
macOS	N/A	N/A	**Yes**	N/A	N/A	**Yes**	No
Windows	**Yes**	**Yes**	**Yes**	N/A	N/A	N/A	No
UWP	N/A	**Yes**	N/A	N/A	N/A	N/A	No

N/A = not applicable

4.1.1 Android

- Supported systems are arm-v7a, arm64-v8a and x86
- Under the hood we're using the Android MediaPlayer API and ExoPlayer 2.8.4 API
- This plugin requires a minimum Android API level of 15 when using the MediaPlayer API, and API level 16 when using ExoPlayer (due to its use of MediaCodec).
- If you want to support streaming don't forget to set the "Internet Access" option in Player Settings to "require"
- For rendering we supports OpenGL ES 2.0 and OpenGL ES 3.0
- Multi-threaded rendering is supported
- The only 3rd-party libraries used are:
    - ExoPlayer 2.8.4
      https://github.com/google/ExoPlayer
    - Facebook Audio 360 1.6.0
      https://facebook360.fb.com/spatial-workstation/

4.1.2 iOS / tvOS

- Supported systems are armv7, arm64
- The iOS simulator is supported on x86 and x86_64
- Under the hood we're using the AVFoundation API
- If you want to support streaming you need to enable HTTP downloads explicitly. For iOS this is an option in newer versions of Unity, but for Mac OS X and older versions of Unity you have to do this explicitly by editing the plist file. There are notes below on how to do this.
- For rendering we support OpenGL ES 2.0, OpenGL ES 3.0 and Metal
- Multi-threaded rendering is supported

- The only 3rd-party libraries used in the iOS/tvOS binaries are:
    - miniz
      https://github.com/richgel999/miniz

4.1.3 macOS

- Only 64-bit (x86_64) builds are supported (Apple dropped 32-bit support back in 2012 when launching OS X 10.8).
- Under the hood we're using the AVFoundation API
- If you want to support streaming you need to enable HTTP downloads explicitly. For iOS this is an option in newer versions of Unity, but for Mac OS X and older versions of Unity you have to do this explicitly by editing the plist file. There are notes below on how to do this.
- For rendering on macOS we support OpenGL Legacy and OpenGL Core and Metal.
- Multi-threaded rendering is supported
- The only 3rd-party libraries used in the macOS binaries are:
    - Hap
      https://github.com/Vidvox/hap
    - miniz
      https://github.com/richgel999/miniz

4.1.4 Windows Desktop

- Under the hood we're using the Media Foundation and DirectShow API's. Media Foundation is used for Windows 8 and above while DirectShow is used as a fallback for Windows 7 and below.
- For rendering we support Direct3D 9, Direct3D 11 and OpenGL Legacy.
- Multi-threaded rendering is supported.
- Windows N / KN edition notes:
    - There are some editions of Windows (N and KN) that ship with greatly reduced built-in media playback capabilities.
    - It seems like these editions don't include MFPlat.DLL, but do include some basic DirectShow components. This means the Media Foundation playback path will not work.
    - These editions of Windows require either a 3rd party codec installed (such as the LAV Filters for DirectShow), or the Microsoft Media Feature Pack:
        - Media Feature Pack for Windows 7 SP1
          https://www.microsoft.com/en-gb/download/details.aspx?id=16546
        - Media Feature Pack for Windows 8.1
          https://www.microsoft.com/en-gb/download/details.aspx?id=40744
        - Media Feature Pack for Windows 10
          https://www.microsoft.com/en-gb/download/details.aspx?id=48231
    - We found that MJPEG DirectShow codec still works on these editions without installing the Media Feature Pack
- The only 3rd-party libraries used in the Windows Desktop binaries are:
    - Hap

https://github.com/Vidvox/hap
    - Google Snappy
      https://github.com/google/snappy
    - GDCL Mpeg-4
      https://github.com/roman380/gdcl.co.uk-mpeg4
    - GLEW
      http://glew.sourceforge.net/
    - Facebook Audio 360 1.6.0
      https://facebook360.fb.com/spatial-workstation/

4.1.5 Windows Store / UWP / Hololens

- For best compatibility and performance add

  appCallbacks.AddCommandLineArg("-force-d3d11-no-singlethreaded");

  To your MainPage.xaml.cs/cpp or MainPage.cs/cpp. You should call this before the appCallbacks.Initialize() function.
- For streaming video don't forget to enable the "InternetClient" capability option in Unity's Player Settings. If you're streaming video from a local server / LAN then you need to enable the "PrivateNetworkClientServer" option.
- No 3rd-party libraries are used in the WSA / UWP binaries

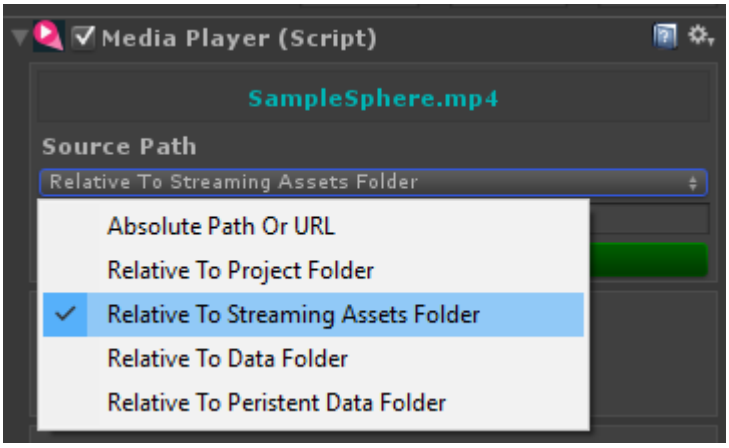
4.1.6 WebGL

- The supported formats and features is dependant on the web browser capabilities
- For best compatibility you can always force WebGL 1.0 instead of 2.0 which is the default. This is done by going to Player Settings > Other Settings > Auto Graphics API and removing WebGL 2.0. We have tested successfully with the following browsers
    - macOS
        - Safari 9.1 (WebGL 1.0 only)
        - Safari 10 (WebGL 1.0 only)
        - Safari 11.0.1 (WebGL 1.0 only)
        - We've seen issues with newer versions of Safari (12, 13 etc) as it doesn't seem to want to play videos that hasn't been set to play by a browser user interface interaction, and Apple seem to be moving away from WebGL to their WebGPU platform in general so are unlikely to support WebGL 2.0
        - Mozilla Firefox
        - Google Chrome
    - Windows
        - Microsoft Edge 38.14393.0.0
        - Mozilla Firefox 51.0
        - Google Chrome 56.0 - 62.0
- The following browsers are not supported:
    - Internet Explorer 11 (any version), instead use the Microsoft Edge browser

4.2 Video File Location

Video files can be played in almost any location, however we recommend placing video files in the **/Assets/StreamingAssets/** folder in your Unity project as this is the easiest folder to get started with. StreamingAssets is a special folder that Unity copies to the build without processing. Files copied elsewhere will require manual copying to the build location.

The MediaPlayer component allows you to browse for video files and specify them relative to a parent folder:



The Video Location field specifies the master location of the video file while the Video Path field specifies where to locate the file relative to the Location.

For example if your file is stored in “Assets/StreamingAssets/video.mp4” you would set the Location to “Relative To Streaming Assets Folder” and set the Video Path to “video.mp4”.

Sub-folders are also supported so a video located at “Assets/StreamingAssets/myfolder/video.mp4” would have it’s Video Path set to “myfolder/video.mp4”.

You can also specify absolute paths, URLs or paths relative to other locations:

4.2.1 Relative To StreamingAssets Folder

This is the best and most common location for video files. This folder is located at “Assets/StreamingAssets/” and you must create it if it doesn’t exist. Files copied to this folder will not be imported or processed by Unity but they will be copied with the build automatically.

On Android though this folder isn’t ideal for massive files, as they are memory mapped and so we’ve seen some devices have memory issues. For Android this folder is fine for small-medium files (up to about 800MB depending on the device types you’re targeting), but beyond that it’s better to copy or download the file into the persistent data folder and load it from there.

4.2.2 Absolute Path or URL

Here you can specify a full URL or absolute path to the video file. A URL could be in the form “http://myserver.com/myvideo.mp4” or “rtsp://myserver.com:8080/mystream.rtsp” depending on the platform support and streaming service used.

An absolute path would look like:

- *C:/MyFolder/AnotherFolder/MyVideo.mp4 (Windows)*
- */Users/Mike/downloads/MyVideo.mp4 (Mac/Linux)*
- */Storage/SD/Videos/MyVideo.mp4 (Android external SDCARD)*
- /Storage/emulated/0/MyFolder/MyVideo.mp4 (Android local file system)
- /mnt/sdcard/MyFolder/MyVideo.mp4 (Android Oculus Go)

Using absolute paths can be useful for testing but isn’t useful when deploying to other machines that don’t necessarily have the same file structure.

4.2.3 Relative To Project Folder

The project folder is the folder of your Unity project, so the folder containing the Assets, Library and Project Settings sub-folders. Specifying files relative to the project folder can be useful when you don’t want to include the video files in your Unity Assets folder but want to keep them within the project folder structure. Often making a sub-folder called “Videos” is useful. One possible problem of using this location is that when making a build your video files will not be copied automatically to the build destination so they require manual copying. For builds this folder should be located:

- Windows - at the same level as your EXE
- Mac - at the same level as the Contents folder in your app bundle
- iOS - at the same level as the AppName.app/Data folder
- Android - not accessible due to APK packaging unless you build the APK manually.

4.2.4 Relative To Data Folder

The data folder is specified by Unity here:
http://docs.unity3d.com/ScriptReference/Application-dataPath.html

It isn’t that useful to put video files into this folder directly as they would then be processed by Unity into VideoClip’s or MovieTexture’s and will bloat your project size. If you want to stop Unity processing the video files simply rename the extension to something Unity doesn’t understand, so “myvideo.mp4” could be renamed to “myvideo.mp4.bin”. Files within the data folder (Assets folder in the editor) are not copied automatically to builds so you would have to manually copy them.

4.2.5 Relative to Persistent Data Folder

The persistent data folder is specified by Unity here:
http://docs.unity3d.com/ScriptReference/Application-persistentDataPath.html

For UWP platforms this would resolve to “ms-appdata:///local/”

4.3 Streaming Notes

AVPro Video supports several streaming protocol depending on the platform:

	HTTP Progressive Streaming	HLS	MPEG-Dash	RTSP
Windows Desktop	Yes	Yes (Windows 10 only)	Yes (Windows 10 only)	Only with ASF stream, or with DirectShow using suitable filter
UWP	Yes	Yes (UWP 10 only)	Yes (UWP 10 only)	No
macOS	Yes	Yes	No	No
iOS	Yes	Yes	No	No
tvOS	Yes	Yes	No	No
Android	Yes	Yes, but better on newer versions	Yes, with ExoPlayer API	Yes, with MediaPlayer API
WebGL	Yes*	Browser specific* **	Browser specific* **	No

* Remember for WebGL streaming you need to have proper CORS set up when accessing other servers / ports. See our notes below on streaming with WebGL.

** In WebGL you can use the hls.js and dash.js libraries. See Streaming section for implementation details.

HTTP Progressive Streaming

When encoding MP4 videos for streaming make sure they are encoded with the video header data at the beginning of the file. You normally do this by selecting “Fast Start” in QuickTime encoder, or use the “-movflags faststart” in FFMPEG, Other encoders will have a similar option. To prepare an MP4 for streaming using FFMPEG you can use the following command:

```
ffmpeg -i %1 -acodec copy -vcodec copy -movflags faststart %1-streaming.mp4
```

Vimeo Note:

If you are streaming videos from VIMEO as MP4 then you should note that you can replace the ".mp4" part in the URL with ".m3u8" to instead make it an HLS stream. This may be particularly useful if you are developing apps for the Apple's App Store as you would need to use HLS streaming to pass certification (as for April 2016). There is also an official Unity plugin for Vimeo (by Vimeo) that integrates with AVPro Video.

4.3.1 macOS, iOS and tvOS Streaming

This platform supports streaming of HLS streams which typically end with the m3u or m3u8 extension.

If you have an HTTPS URL it should work fine because Apple trusts the secure connection.

If you can only use HTTP then your app has to have a special flag set to let it use HTTP connections (this is a security issue for Apple).

This setting is exposed in the Unity Player Settings here for iOS and tvOS:

The setting is also exposed in the scripting API here:
http://docs.unity3d.com/ScriptReference/PlayerSettings.iOS-allowHTTPDownload.html

If for some reason your version of Unity doesn't expose this then you will have to add it manually. In the Unity editor you need to edit "Unity.app/Contents/Info.plist" and in your built application you would need to edit "your.app/Contents/Info.plist". These files need to have these keys added:

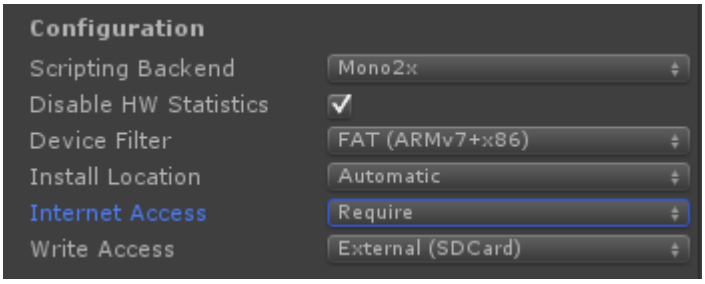
```
<key>NSAppTransportSecurity</key>
<dict>
 <key>NSAllowsArbitraryLoads</key>
 <true/>
</dict>
```

You can find more information about this here:
http://ste.vn/2015/06/10/configuring-app-transport-security-ios-9-osx-10-11/

We've also included a post process build script called "PostProcessBuild.cs" in the project which edits the plist and adds this attribute. Currently it's only set for iOS but you can edit the #define at the top to allow Mac OS X too.

4.3.2 Android Streaming

Using the ExoPlayer API is recommended for streaming video as it generally has wider support for streaming protocols. Android streaming requires the Internet Access setting (in Player Settings) to be set to “Require”:



Also note that starting with Android 9 (API level 28) cleartext support (unencrypted connections) is disabled by default, which can cause some HTTP streams to fail. You should be able to resolve this by switching the URL to HTTPS or by adding android:usesCleartextTraffic="true" into your AndroidManifest.xml file

4.3.3 UWP / Windows Phone / Hololens Streaming

Make sure to tick the “InternetClient” capabilities option in Player Settings. If you’re streaming video from a local server / LAN then you need to enable the “PrivateNetworkClientServer” option.

4.3.4 WebGL Streaming

If you are trying to access a URL on another server/port/domain then you need to have CORS (cross-origin resource sharing) configured on that server to allow access. Websites like https://enable-cors.org/ show you how to configure CORS on different web servers. If you are hosting on a S3 bucket there are also ways to configure this. You can also test whether CORS is the issue by installing a browser plugin to toggle CORS, for example this one for Chrome: https://chrome.google.com/webstore/detail/allow-control-allow-origi/nlfbmbojpeacfghkpbjhddihlkkiljbi

HLS and MPEG-DASH are not natively supported by all browsers. We have added hooks to include third-party javascript libraries to handle these formats. Under the “Platform Specific > WebGL” section you can select “External Library”. This will force the MediaPlayer to use either hls.js or dash.js. You can also select “custom” if you wish to add support for your own javascript library.

To add support or dash.js:

1. Download the latest dash.js release (we tested with 2.8.0): https://github.com/Dash-Industry-Forum/dash.js/releases
2. Copy “dash.all.min.js” to the Assets/Plugins/WebGL folder and rename it “dash.all.min.jspre”
3. In the MediaPlayer component set Platform Specific > WebGL > External Library to dash.js
4. Build for WebGL

To add support for hls.js:

1. In the MediaPlayer component set Platform Specific > WebGL > External Library to hls.js
2. Build for WebGL
3. Download the latest hls.js release (we tested with 0.10.1): https://github.com/video-dev/hls.js/releases
4. Once your build is made, copy “hls.min.js” to the TemplateData folder
5. Edit the index.html to add <script src="TemplateData/hls.min.js"></script> before the UnityLoader.js script is loaded. Ideally you would add this to a new WebGL template so that you don’t have to make these changes for each build.

4.3.5 Test Streams

We found these streams handy for testing (no guarantee that they’re still working):

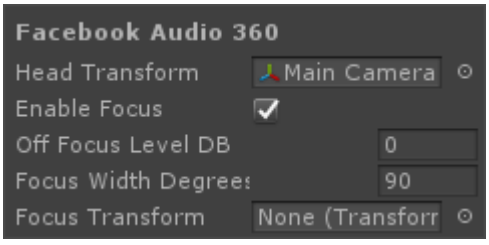
- **Streaming MP4**
  - **HTTP**
    http://downloads.renderheads.com/2016/BigBuckBunny_360p30_Streaming.mp4
  - **HTTPS**
    https://drive.google.com/uc?export=download&id=0B0JMGMGgxp9WMEdWb1hyQUhlOWs
- **HLS**
  - http://qthttp.apple.com.edgesuite.net/1010qwoeiuryfg/sl.m3u8
  - Apple Test streams (from https://developer.apple.com/streaming/examples/)
    - **Basic complexity**
      https://devimages.apple.com.edgekey.net/streaming/examples/bipbop_4x3/bipbop_4x3_variant.m3u8
    - **Medium complexity**
      https://devimages.apple.com.edgekey.net/streaming/examples/bipbop_16x9/bipbop_16x9_variant.m3u8
    - **Advanced TS**
      https://devstreaming-cdn.apple.com/videos/streaming/examples/img_bipbop_adv_example_ts/master.m3u8
    - **Advanced fMP4**
      https://devstreaming-cdn.apple.com/videos/streaming/examples/img_bipbop_adv_example_fmp4/master.m3u8
- **MPEG-Dash**
  - http://rdmedia.bbc.co.uk/dash/ondemand/bbb/2/client_manifest-high_profile-common_init.mpd
  - http://www.bok.net/dash/tears_of_steel/cleartext/stream.mpd
- **RTSP**
  - rtsp://rtmp.infomaniak.ch/livecast/latele
- **RTMP**
  - RTMP is not supported on any platform yet (unless 3rd party support is used)

4.4 Audio Notes

4.4.1 Facebook Audio 360 (Windows & Android only)

Spatial audio support is currently available using Facebook Audio 360 on Windows desktop and Android. On Windows only Windows 10 and above is supported and the Media Foundation video API must be selected. On Android the ExoPlayer video API must be selected. The video files must be using a MKV file container and audio must be using the Opus codec encoded with Facebook Audio 360.

The best way to encode the video is to use the FB360 Encoder tool which comes as part of the FB360 Spatial Workstation. Set Output Format to “FB360 Matroska (Spatial Workstaton 8 channel)” and then set your video and audio source files and encode your video. This should create a MKV file with 10 channels of Opus audio.



The settings are located under the Audio section of the MediaPlayer component. The “Head Transform” field must be set to the transform that represents the player's head so that rotation and positional changes affect the audio rendering. Usually this is the main camera.

“Enable Focus” can be enabled when a specific region of audio in the 360 field needs to be given focus. The rest of the audio has its volume reduced.

Next the Facebook Audio 360 support must be enabled for each platform that needs it via the “Platform Specific” panel. Currently it is only available on Windows desktop and Android.

The “Channel Mode” must be set to the channel encoding mode used when creating the video. Currently this can not be determined automatically. The default is “TBE_8_2” which means 8 channels of hybrid ambisonics and 2 channels of head-locked stereo audio.
More information on encoding etc can be found on the Facebook Audio 360 website at:
https://facebook360.fb.com/spatial-workstation/

Alternative steps for encoding manually

1. Create a WAV file with the audio format they need (Eg 9 channels ambisonics with 2 channels of head-locked audio will require a 11 channel WAV file with the 2

head-locked channels at the end)

2. Use Opus tools, as described here to convert the WAV file to Opus:
   https://opus-codec.org/downloads/
   https://facebookincubator.github.io/facebook-360-spatial-workstation/Documentation/SDK/Audio360_SDK_GettingStarted.html#encoding-opus-files

3. Use ffmpeg to mux this opus file into the video container (ensure that the video file doesn't have any audio first):
   ffmpeg -i audio.opus video.mp4 -c:a copy -c:v copy audio_video.mkv

4. In AVPro Video specify the required channel map

Converting existing ambisonic videos

It is also possible to convert existing ambisonic videos so they are compatible. For example if you have an existing MP4 file with 4-channel 1st order ambisonic audio, then it is possible to convert this into the above format (MKV container with Opus audio) using a tool like FFMPEG. Simple put the following command in a .BAT file and then drag your MP4 into the batch file:

ffmpeg -y -i input.mp4 -c:v copy -acodec libopus -mapping_family 255 output.mkv

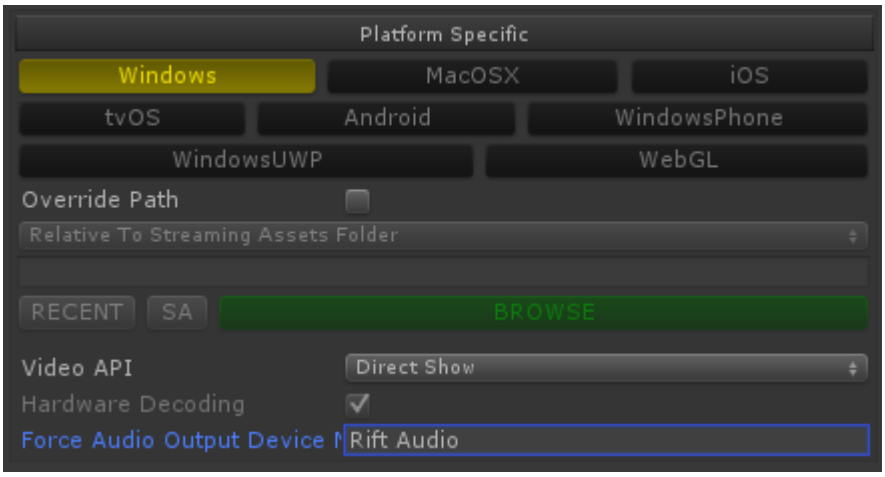
This should then generate an MKV file that you can play with AVPro Video. All that remains is to set the channel mapping in the MediaPlayer component to AMBIX_4.

4.4.2 Audio Device Redirection (Windows only)

Currently only the Windows plugin has support for audio manipulation beyond the standard volume and stereo panning.

Some VR systems such as the Oculus Rift and HTC Vive have their own audio output device and one needs to redirect audio to these devices instead of the system default audio device. Unity does this automatically for its internal audio, but AVPro Video renders to the system default audio device.

This issue can be solved by either using the AudioOutput component (see next section). This component redirects the audio to be rendered by Unity and so it goes to the correct device. AudioOutput requires that the Media Foundation video API be used, so if you need to use the DirectShow API you can specify the name of the output device manually in the field "Force Audio Output Device":



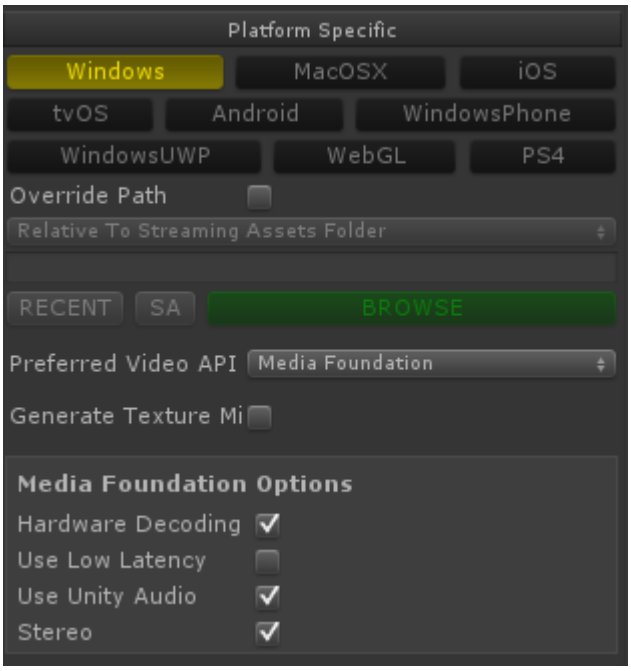
The device name to use can be retrieved from the VR API or hard coded. For Oculus Rift the name is usually “Rift Audio” or “Headphones (Rift Audio)” and for HTC Vive it is “HTC VIVE USB Audio”. The Facebook Audio 360 audio output option on Windows also allows you to specify a specific audio output device name, but this requires that the your audio is encoded with the Opus codec.

4.4.3 Audio Spatialisation (Windows only)

Audio needs to rotate as the user moves their head in the virtual world. This can be achieved by using the AudioOutput component which redirects the audio from the video into Unity (via Unity’s AudioListener).

This component should be stacked above its required AudioSource component.

AudioOutput requires that the video API is set to Media Foundation and that the “Use Unity Audio” tickbox is selected.

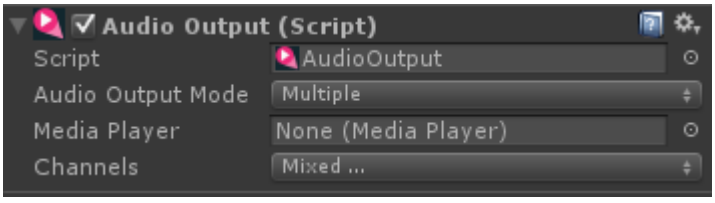


For proper 3D audio placement you will also need to add a spatialiser plugin to the Unity Audio Settings screen (in Unity 5.4 and above) and tick the “spatialize” tickbox on the Audio Source. Multiple instances of AudioOutput component can be created, each one outputting a different audio channel from a different world position.

If Unity Audio is enabled but the Stereo checkbox is not selected, AVProVideo will try to detect how many channels the video has, and send to Unity as many channels as requested. To get all audio channels, make sure that the default speaker mode in the Unity Audio Settings supports enough channels, and ensure that the OS audio sound settings are in the correct speaker mode (otherwise Unity will just default to the maximum number of supported channels). If the Stereo box is checked, the audio will get resampled to 2 channels.

If you wish to only get specific channels from the video, you can choose to mask the channels using the AudioOutput component.

4.4.4 Audio Channel Remapping (Windows only)



The AudioOutput and AudioChannelMixer components allow the ability to route different audio channels from the media to different physical channels. If you set Audio Output Mode to Single then the audio from a specific channel will be duplicated to all other channels. For best results make sure your video only has mono audio. You can then use the AudioChannelMixer component to control which speaker the audio goes to.

4.5 Augmented / Virtual Reality Notes

So far we have tested AVPro Video with:

- Gear VR
- Google Cardboard
- Google Daydream
- Oculus Rift
- HTC Vive
- Microsoft Hololens

VR is still very new and you should always check for the latest recommended installation steps when creating your project. We found a lot of out of date setup instructions on the net.

AVPro Video supports 4K MP4 playback for creating 360 degree experiences. Stereo 4K videos in both top-bottom and side-by-side formats are also supported. If you're using Windows 10 and have an Nvidia Geforce 10xx series (eg 1070) you can display 8K H.265 videos (requires 64-bit build).

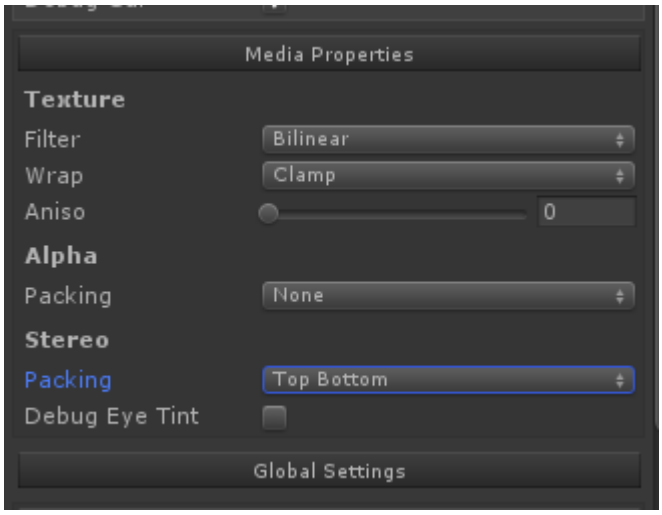
See the FAQ for tips to achieve high resolution video playback for VR.

For software decoders reducing the complexity of the encoded video will give the decoding engine a much easier time and could result in higher frame rates and lower CPU/GPU usage. Possible encoding tweaks include:

- Use the lowest profile level possible
- Don't use too many reference frames
- Don't use too many b-frames
- Disable CABAC
- Use the slices option (eg -slices 4)

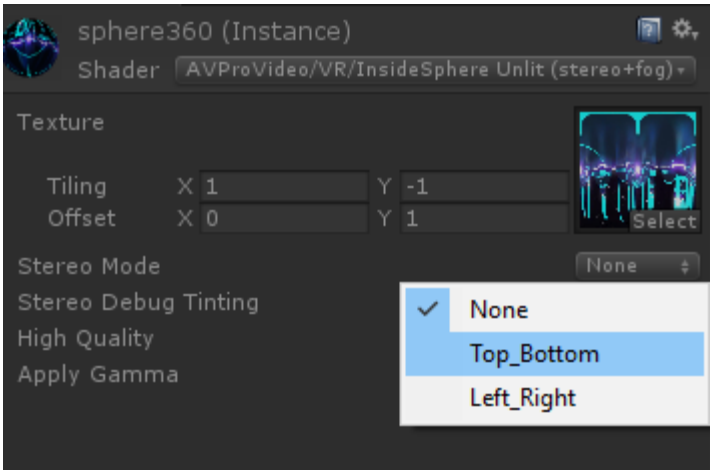
4.5.1 Stereo VR

AVPro Video supports stereoscopic videos in the top-bottom and left-right formats. You can set the stereo packing format of your video in the Media Properties panel:



Now when using the InsideSphere shader on a mesh it will automatically map the right part of the video to each eye. See the “Demo_360SphereVideo” scene for an example of how this works.

Optionally you can manually set the material properties. The included shader “InsideSphere.shader” allows you to easily set what format your video is in via a drop-down in the material:



Select “Stereo Debug Tinting” to colour the left and right eyes different colours so you can be sure the stereo is working.

NOTE: Be sure to add the "UpdateStereoMaterial” component script to your scene when using this material and a stereo video. Often stereo VR requires 2 cameras, each set to a different layer mask and 2 spheres also set to a different mask. AVPro Video doesn’t require this and just uses your normal single camera and single sphere.

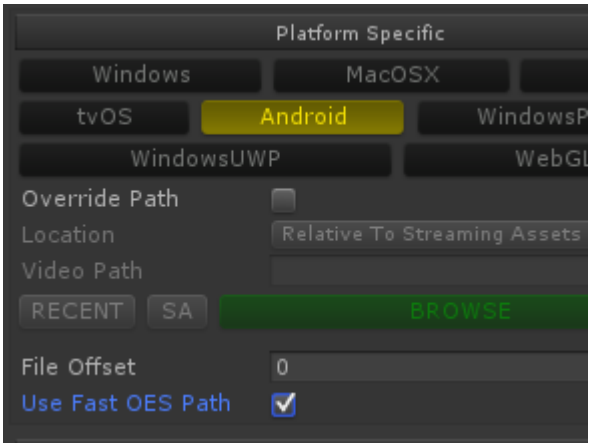
NOTE: If you’re playing stereo videos on Windows and the stereo isn’t appearing properly, please read our FAQ #8.

An example of how to pack 2 videos together into a left-right stereo packed layout using FFMPEG:

```
ffmpeg -i left.mp4 -vf "[in] pad=2*iw:ih [left]; movie=right.mp4
[right];[left][right] overlay=main_w/2:0 [out]" stereo.mp4
```

4.5.2 Android OES playback path

For Android there is a special playback option called “Use Fast OES Path”. This option caters especially for VR where users are trying to get the highest possible frame rate and resolution out of the device (without it overheating at the same time). The option is available in the Platform Specific section of the MediaPlayer component:



The OES path is not enabled by default because it requires some special care to be taken and can be tricky for beginners. When this option is enabled the Android GPU returns special OES textures (see EGL extension OES_EGL_image_external) that are hardware specific. Unfortunately Unity isn’t able to use these textures directly, so you can’t just map them to a material or UI. To use the texture a GLSL shader must be used. Unfortunately Unity’s GLSL support isn’t as good as its CG shader support so again this makes things more tricky. The GLSL compiler only happens on the device (not inside Unity) so errors in the shader can be difficult to debug.

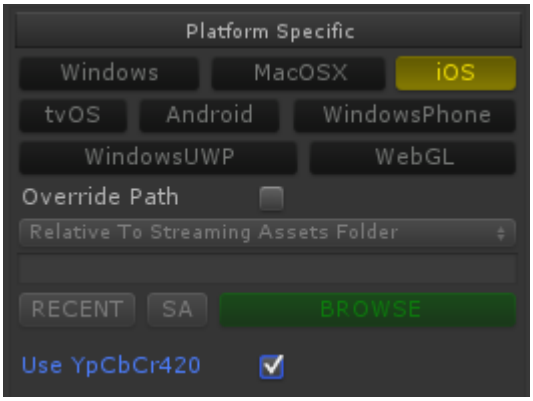
We have included a version of the VR sphere shader that supports stereo videos as an example. Hopefully in the future we can improve the integration of these shaders so they aren’t such special cases. This playback path is much faster though, so is definitely worth exploring. Note that for VR stereo rendering, OES only currently supports multi-pass rendering path, and not single-pass or single-pass instanced.

For the sphere demo scene, simply change the shader on the sphere mesh material to be one of the “VR” “OES” ones and tick the “Use Fast OES Path” on the MediaPlayer component.

4.5.3 iOS YCbCr playback path

For iOS and tvOS we have added support for YCbCr textures which results in memory saving compared to standard RGBA32 textures. This option is enabled by default but can

be disabled on the MediaPlayer here:



This option is ideal for memory saving which is especially important when targeting low-end devices (with 1GB RAM) and when playing back very high resolution video, such as 4K for VR content. Improved performance may also be experienced using this option. The DisplayIMGUI and DisplayUGUI components automatically detect this and switch to a suitable shader. ApplyToMesh/ApplyToMaterial also detect this setting and tries to set up the shader on the material to the correct settings, however it requires the shader to have the correct properties. The AVPro Video shaders support this, so if you want to use this on a mesh then make sure you're using these shaders.

4.6 Hap Codec Notes

The Hap video codec is natively supported by AVPro Video on certain platforms and has the following benefits:

- Very low CPU usage
- GPU decompression
- Low memory usage
- Supports very high resolutions
- Supports alpha channel transparency

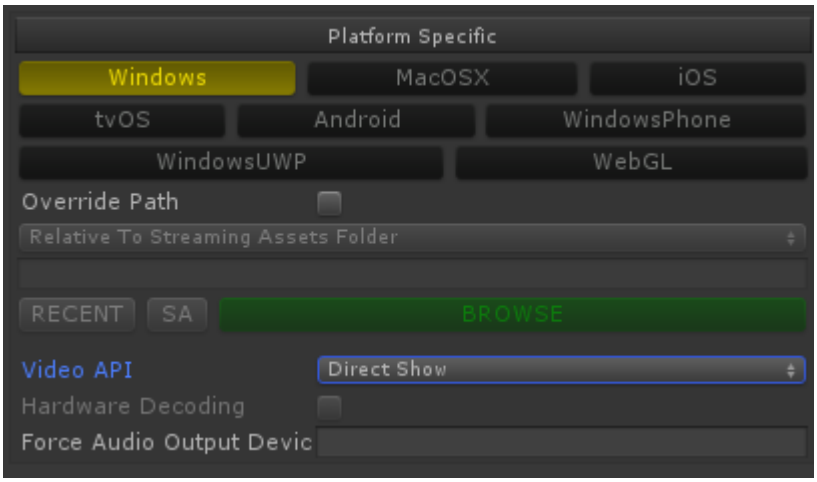
The main downside is:

- Very large files

AVI and MOV containers can both be used however we recommend the MOV container. Hap is only supported on Windows and Mac OS X platforms.

4.6.1 Windows Support

Hap, Hap Alpha, HapQ and HapQ Alpha are supported. Hap currently requires the "DirectShow" video API to be selected:



4.6.2 macOS Support

Hap, Hap Alpha, HapQ and HapQ Alpha are supported.

4.6.3 Encoding

You can download the QuickTime codec for Windows and macOS here:
https://github.com/Vidvox/hap-qt-codec/releases

This codec can be used with QuickTime Pro or any other software that supports QuickTime codecs such as Adobe After Effects and Adobe Premiere.

Alternatively you can use a recent build of FFMPEG with the following command-lines:

- `ffmpeg -i input.mov -vcodec hap -format hap output-hap.mov`
- `ffmpeg -i input.mov -vcodec hap -format hap_alpha output-hap.mov`
- `ffmpeg -i input.mov -vcodec hap -format hap_q output-hap.mov`

Notes:

- You can also add the “-chunks 4” option which will encode each frame into 4 chunks so the decoding work can be split across multiple threads, resulting in faster decoding as long as the disk can keep up.
- Width and height must be multiple of a 4.
- Hap Alpha requires straight not pre-multipled alpha.
- Sadly ffmpeg doesn’t yet support the HapQ Alpha format.
- We don’t support Hap Q Alpha variant in Windows when using the legacy D3D9 graphics API

4.7 Transparency Notes

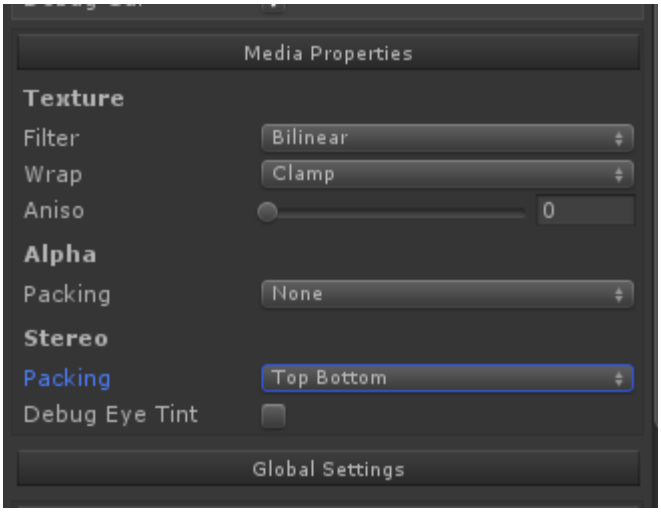
Not many video codecs have native support for transparency / alpha channels. Formats supported by some platforms of AVPro Video are:

- HEVC+Alpha

    - Requires macOS 10.15, iOS 13.0 or tvOS 13.0
- Hap Alpha
    - Great support on Windows and macOS. Fast and low overhead format, though file size can get large depending on the content. Currently this is the format we recommend for transparent video.
- Hap Q Alpha
    - Great support on Windows and macOS. Slightly higher quality and file size compared to Hap Alpha.
- Uncompressed RGBA
- Uncompressed YUVA
    - Uncompressed isn't ideal for file size or disk bandwidth but can still be used as a fallback
- ProRes 4444
    - Best support is on macOS. Files can be huge though.
- VP6
    - Legacy format. We support it only via 3rd party DirectShow plugins for Windows (eg LAV Filters)

4.7.1 Alpha Packing

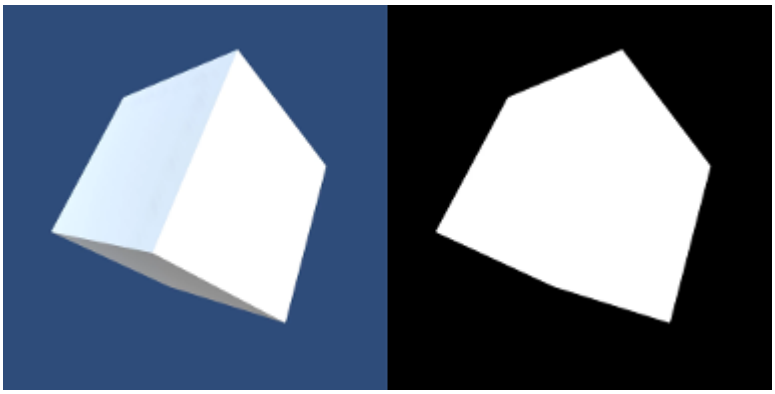
Alternatively you can encode your videos in video formats that don't support an alpha channel by packing the alpha channel into the same frame. You can double the width for a left-right packing layout, or double the height for a top-bottom packing layout. This packing could be created in software such as AfterEffects, or the command-line ffmpeg tool can be used. The packing format is set in the "Media Properties" panel of the AVPro Video MediaPlayer component:



Here we show two examples using ffmpeg to convert a source video containing a transparency/alpha channel into an alpha packed format:

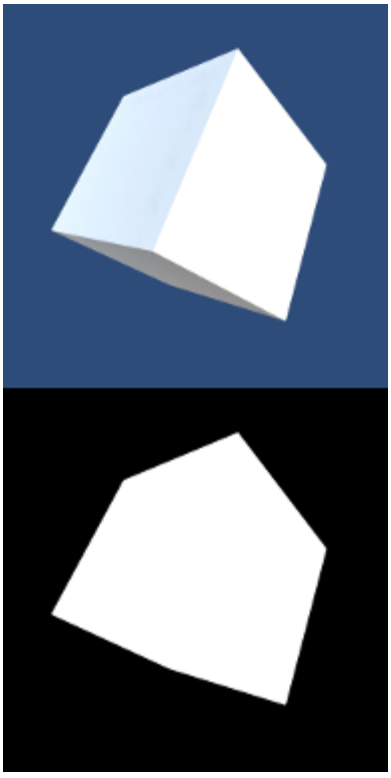
Left-right alpha packing:

```
ffmpeg -i %1 -vf "split [a], pad=iw*2:ih [b], [a] alphaextract, [b]
overlay=w" -y %1.mp4
```



Top-bottom alpha packing:

```
ffmpeg -i %1 -vf "split [a], pad=iw:ih*2 [b], [a] alphaextract, [b]
overlay=0:h" -y %1-tb.mp4
```



4.8 Hardware Decoding

AVPro Video supports hardware decoding on most platforms for optimal performance. WebGL, Windows 8.1, Windows 10, macOS, tvOS, iOS and Android all default to hardware decoding when possible. When playing back multiple videos simultaneously one must be careful not to exceed the number of videos that the hardware can play back smoothly. We have tried to collect information about the limits of specific hardware.

4.8.1 NVidia

NVidia GPU's use a technology called "Purevideo" or "NVDec" to off-load decoding from the CPU. More information can be found here:
https://en.wikipedia.org/wiki/Nvidia_PureVideo#Nvidia_VDPAU_Feature_Sets

Some NVidia Purevideo capabilities:

KEPLER (GK107, GK104)	MAXWELL 1 (GM107, GM204, GM200)	MAXWELL 2 (GM206)	PASCAL (GP100)
MPEG-2, MPEG-4, H.264	MPEG-2, MPEG-4, H.264, HEVC with CUDA acceleration	MPEG-2, MPEG-4, H.264 HEVC/H.265 fully in hardware	MPEG-2, MPEG-4, H.264 HEVC/H.265 fully in hardware
H.264: ~200 fps at 1080p; 1 stream of 4K@30	H.264: ~540 fps at 1080p 4 streams of 4K@30	H.264: ~540 fps at 1080p 4 streams of 4K@30	?
H.265: Not supported	H.265: Not supported	H.265: ~500 fps at 1080p 4 streams of 4K@30	?

(table adapted from presentation "HIGH PERFORMANCE VIDEO ENCODING WITH NVIDIA GPUS")

GPU Architecture	MPEG-2	VC-1	H.264/ AVCHD	H.265/ HEVC	VP8	VP9
Fermi (GF1xx)	Maximum Resolution: 4080x4080	Maximum Resolution: 2048x1024 1024x2048	Maximum Resolution: 4096x4096 Profile: Baseline, Main, High profile up to Level 4.1	Unsupported	Unsupported	Unsupported
Kepler (GK1xx)	Maximum Resolution: 4080x4080	Maximum Resolution: 2048x1024 1024x2048	Maximum Resolution: 4096x4096 Profile: Main,	Unsupported	Unsupported	Unsupported

			High profile up to Level 4.1			
Maxwell Gen 1 (GM10x)	Maximum Resolution: 4080x4080	Maximum Resolution: 2048x1024 1024x2048	Maximum Resolution: 4096x4096 Profile: Baseline, Main, High profile up to Level 5.1	Unsupported	Unsupported	Unsupported
Maxwell Gen 2 (GM20x)	Maximum Resolution: 4080x4080	Maximum Resolution: 2048x1024 1024x2048 Max bitrate: 60Mbps	Maximum Resolution: 4096x4096 Profile: Baseline, Main, High profile up to Level 5.1	Unsupported	Maximum Resolution: 4096x4096	Unsupported
Maxwell Gen 2 (GM206)	Maximum Resolution: 4080x4080	Maximum Resolution: 2048x1024 1024x2048 Interlaced	Maximum Resolution: 4096x4096 Profile: Baseline, Main, High profile up to Level 5.1	Maximum Resolution: 4096x2304 Profile: Main profile up to Level 5.1	Maximum Resolution: 4096x4096	Maximum Resolution: 4096x2304 Profile: Profile 0
Pascal (GP100)	Maximum Resolution: 4080x4080	Maximum Resolution: 2048x1024 1024x2048	Maximum Resolution: 4096x4096 Profile: Baseline, Main, High profile up to Level 5.1	Maximum Resolution: 4096x4096 Profile: Main profile up to Level 5.1	Maximum Resolution: 4096x4096	Maximum Resolution: 4096x4096 Profile: Profile 0
Pascal (GP10x)	Maximum Resolution: 4080x4080	Maximum Resolution: 2048x1024 1024x2048	Maximum Resolution: 4096x4096 Profile: Baseline, Main, High profile up to Level 5.1	Maximum Resolution: 8192x8192 Profile: Main profile up to Level 5.1	Supported2 Maximum Resolution: 4096x4096	Maximum Resolution: 8192x8192 Profile: Profile 0

(Table adapted from Nvidia Video Decoder Interface documentation: https://developer.nvidia.com/nvdec-programming-guide)

4.8.2 AMD

AMD UVD
https://en.wikipedia.org/wiki/Unified_Video_Decoder

UVD Version	Hardware Code Names
UVD 1.0	RV610, RV630, RV670, RV620, RV635
UVD 2.0	RS780, RS880, RV770

UVD 2.2	RV710, RV730, RV740
UVD 2.3	CEDAR, REDWOOD, JUNIPER, CYPRESS
UVD 3.0	PALM (Wrestler/Ontario), SUMO (Llano), SUMO2 (Llano
UVD 3.1	BARTS, TURKS, CAICOS, CAYMAN
UVD 3.2	ARUBA (Trinity/Richland), TAHITI
UVD 4.0	CAPE VERDE, PITCAIRN, OLAND
UVD 4.2	KAVERI, KABINI, MULLINS, BONAIRE, HAWAII
UVD 5.0	TONGA
UVD 6.0	CARRIZO, FIJ
UVD 6.2	STONEY
UVD 6.3	POLARIS10, POLARIS11, POLARIS12

(table adapted from X.org RadeonFeature wiki:
https://www.x.org/wiki/RadeonFeature/#index8h2)

UVD Version	MPEG 2	MPEG 4	MPEG-4 AVC/VC1	HEVC	Max Size	Notes
UVD 1.0	No	No	Yes	No	2K	
UVD 2.0	No	No	Yes	No	2K	
UVD 2.2	No	No	Yes	No	2K	
UVD 2.3	No	No	Yes	No	2K	
UVD 3.0	Yes	Yes	Yes	No	2K	
UVD 3.1	Yes	Yes	Yes	No	2K	
UVD 3.2	Yes	Yes	Yes	No	2K	
UVD 4.0	Yes	Yes	Yes	No	2K	
UVD 4.2	Yes	Yes	Yes	No	2K	
UVD 5.0	Yes	Yes	Yes	No	4K	
UVD 6.0	Yes	Yes	Yes	Yes	4K	
UVD 6.2	Yes	Yes	Yes	Yes	4K	Supports 10bit
UVD 6.3	Yes	Yes	Yes	Yes	4K	Supports 10bit

(table adapted from X.org RadeonFeature wiki:
https://www.x.org/wiki/RadeonFeature/#index8h2)

4.8.3 Intel

Intel's Quick Sync technology is a dedicated video decoder hardware built into the CPU.

Intel Kaby Lake supports H.265 10-bit decoding!

4.9 Multi-GPU SLI / CrossFire Notes

Multiple GPU's are often used to accelerate high-end rendering but it brings with it some subtle challenges for video playback. Here we write some notes about our experience using Nvidia SLI on Windows (specifically Windows 8.1 with dual Nvidia M6000 cards).

Using Alternate Frame Rendering SLI (AFR SLI) can cause stuttering when using the Media Foundation hardware decoding because only one GPU is doing the decoding and then it must pass the frames back to the other GPU which is a slow operation and defeats the purpose of SLI. One option here is to disable GPU decoding in AVPro Video, which is fine for lower resolution videos, but often the CPU just isn't fast enough for very high resolution content. AFR can also suffer from so called "micro stutters" caused by each GPU handling the flip presentation independently which isn't ideal for video playback on large video walls.

Another option is to use the cards in "SLI VR" mode … To be continued..

4.10 Subtitle Notes

AVPro Video supports external subtitles in the SRT format. Subtitles internal to media files are not yet supported. See the subtitle demo scene to see how to use subtitles.

4.11 DRM / Encryption Notes

DRM support is currently not a key feature of AVPro Video, but we aim to improve support in the future. DRM that we know works with the plugin includes:

- Android
    - HLS with AES-128 clear-key (make sure your TS segments are 188 bytes aligned for maximum Android compatibility)

    - We have a file offset feature which allows you to access files hidden within a file at an offset. Not strictly DRM but it can be used as a quick way to hide video files within other data
        - In Windows you can easily append your video to a dummy video file with the following command:

          copy /b DummyVideo.mp4 + %1 %~n1-hidden.mp4

    - Custom HTTP header fields can be specified which can help with server side validation

- macOS & iOS & tvOS
    - HLS with AES-128 clear-key, direct key and key request using an auth token in the HTTP header ("Authorization" field). More information about HLS encryption can be read in the RFC here: https://tools.ietf.org/html/draft-pantos-http-live-streaming-23

    - Custom HTTP header fields can be specified which can help with server side validation

*DRM schemes Fairplay, Widevine, PlayReady etc are not yet supported.

4.11.1 HLS Decryption

On the Apple platforms (macOS, iOS and tvOS) we support HLS streams encoded with AES encryption. Key retrieval from a server URL usually requires an authentication token, which can be specified using the method SetKeyServerAuthToken(string), or this can be ignored if your key retrieval server doesn't require any token for key retrieval (clear-key). The auth token is a string that is inserted into the "Authorization" HTTP field when retrieving the decryption key from the server URL specified in the HLS manifest.

We also added some functionality to override the key URL and to specify the key data directly. SetDecryptionKey(byte[]) and SetDecryptionKeyBase64(string) can be used to specify the key directly. Using this will bypass any server key retrieval. SetKeyServerURL(string) can be used to override the key retrieval URL. Both of these are useful for debugging.

These options can also be specified in the MediaPlayer inspector UI (in the Platform Specific section), and in the PlatformOptions properties, eg mediaPlayer.PlatformOptionsIOS

4.11.2 HTTP Headers

Custom HTTP headers can be useful for many uses. Typically we have seen them used for authentication, token exchange and cookies. Here are Some examples of formats we've used in the past:

For authentication the typical HTTP headers are used

Authorization Basic <username>:<password>
Authorization Bearer <token>

In JSON these would become:

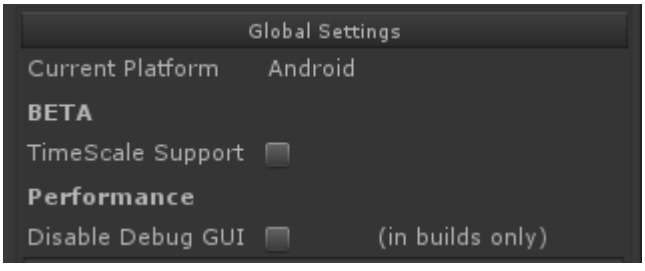
```
{
    "Authorization":"Basic <username>:<password>"
}

{
    "Authorization":"Bearer <token>"
}
```

And with cookies you can pass multiple values, here is a typical JSON format:

```
{
"Cookie":"<cookie-name>=<cookie-value>;<cookie-name2>=<cookie-value2>;"
}
```

4.12 Video Capture Notes



To make a non-realtime video capture of your Unity scenes which include videos, requires the video playback to slow down or speed up to match the video capture rate. AVPro Video supports this through the “TimeScale Support” option which is found in the Global Settings panel of the Media Player component. This means you can create high quality renders running at 1fps to produce a smooth 60fps video, and any videos in your scene will play back at the correct rate for the recording. Audio is not supported though when using this option (as is the case in Unity itself).

4.13 Seeking / Playback Rate Notes

Most videos are optimally encoded for the main use case: normal forward playback with inaccurate seeking.
If you want to start changing the playback rate, play in reverse or have frame accurate seeking then you may run into issues where the playback becomes extremely slow or the seeking is not accurate. There are several reasons for this, but it mostly is related to how the video is encoded and specifically the key-frame distribution. There are also some platform differences to consider.

4.13.1 Video Codecs and Encoding

Codecs such as H.264 and H.265 generally compress video frames so that they depend on data included with previously decoded frames. These are called P and B frames and seeking to one of these is computationally expensive as in order to display them the decoder must first decode the other frames that they depend on. The other type of frame is called a key-frame or I-frame and these frames can be decoded immediately because they don’t depend on any other frames. Compressing using P And B frames is known as temporal compression and isn’t ideal for accurate random seeking or playback rate changes.

For the best results you would to encode your video with only key-frames, as then you can seek accurately and quickly anywhere in the video. This is possible, but increases the file size dramatically. Using FFMPEG you can encode a video to use key-frames only using the “-g 1” option. Another option would be to use a codec that only supports key-frames, such as Hap or ProRes - but again these result in large file sizes and limited GPU decoding capabilities.

In most codec with temporal compression the key-frames are spaced every 250 frames. Some platforms can only seek to the key-frames (see table below), while others can do accurate seeking but this can be very slow if the distances between key-frames is too large. Try reducing the key-frame distance for faster seeking. You can also reduce the decoder complexity by encoding with a fastdecode tuning option.

4.13.2 Playback Rate

Generally we recommend these rates:

0.25, 0.5, 1.0, 1.25, 1.5, 1.75, 2.0

Going up to 4.0 might be possible depending on your platform, machine specs and the codec used. Increasing playback rate usually places more demand on the video decoder and also on the disk/network source, and these limit how much you can increase the playback rate by.

Using negative values isn't generally recommended as it isn't as well supported, but if you do have to use a negative rate then also try keeping the numbers small such as:
-0.25, -0.5, -1.0

Audio also may or may not play when changing the playback rate - this depends on the platform (see table below).

One safe alternative to adjusting rate is to pause the video and fast seek to simulate a change in playback rate. This approach would work on all platforms.

Video encoding can also help the performance of a change in playback rate. Videos with more key-frames (or ideally all key-frames) and with less complex encoding (eg no B frames, CABAC disabled etc) will work better. Alternatively a key-frame-only codec could be used, such as Hap.

4.13.3 Platform Differences

Seeking:

Platform and Video API	Fast Inaccurate Keyframe Seeking	Slow Accurate Frame Seeking	Adjust Playback Rate
Android MediaPlayer API	Yes	API 26 and above	API 6 and above
Android ExoPlayer API	Yes	Yes	Yes
Windows Media Foundation	Yes	Yes	Yes
Windows DirectShow	Yes	Depends on the codec	Yes but not negative
UWP Media Foundation	Yes	Yes	Yes
macOS	Yes	Yes	Yes If source supports it
iOS & tvOS	Yes	Yes	Yes If source supports it
webGL	Yes	Varies	Yes but not negative

Playback Rate:

Platform and Video API	Adjust Playback Rate	Audio Plays	Negative Rates
Android MediaPlayer API	API 23 and above	Yes	?
Android ExoPlayer API	Yes	Yes	?
Windows Media Foundation	Yes	Depends on codec	Yes
Windows DirectShow	Yes	No	No
UWP Media Foundation	Yes	Depends on codec	Yes
macOS	Yes	Yes	Depends on media source
iOS & tvOS	Yes	Yes	Depends on media source
webGL	Yes	Depends on browser	No

5. Quick Start Examples

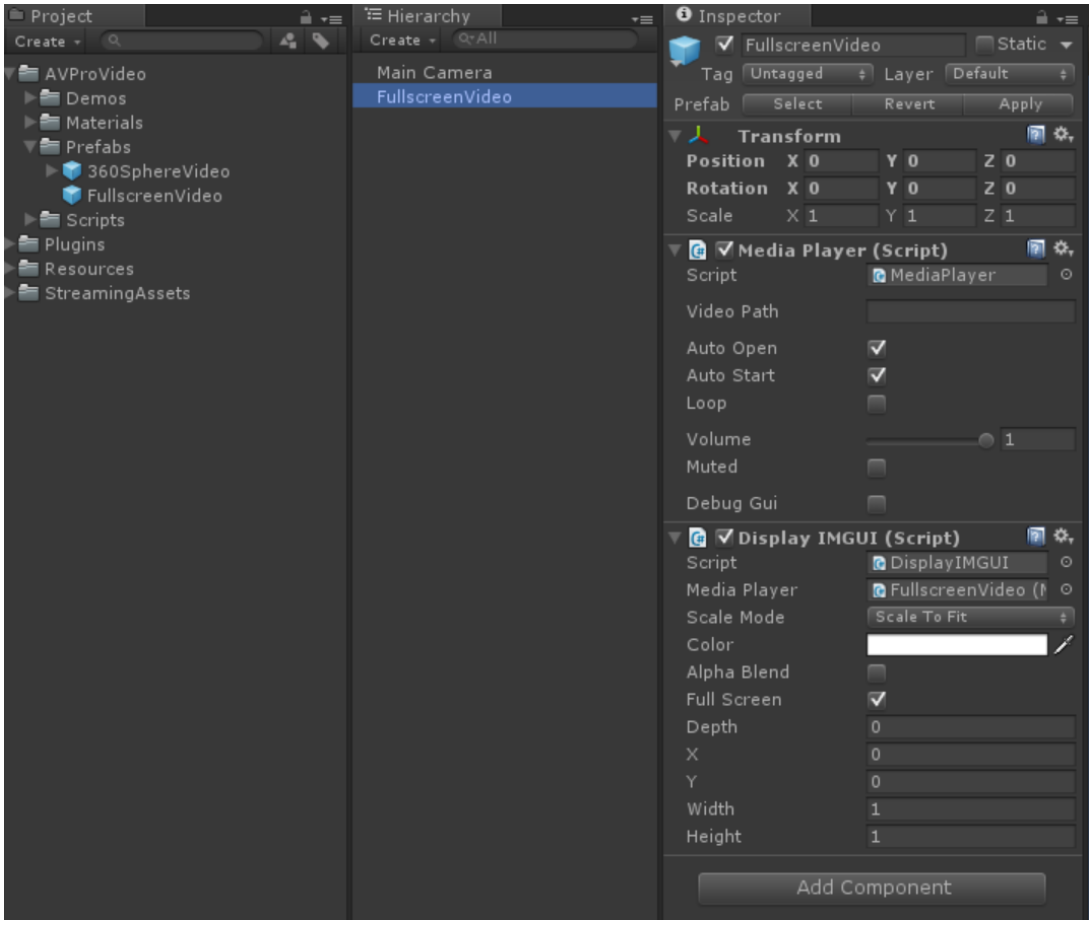
5.1 Quick Start: Fastest Start for Unity Experts

1. Put video files in the StreamingAssets folder
2. Use the MediaPlayer script to play your video (set Video Path to the file name of your video file).
3. Use one of the display scripts to display your video (eg DisplayIMGUI, DisplayUGUI, ApplytoMaterial)

5.2 Quick Start: Fullscreen Video Player using Prefabs

AVPro Video includes a number of example prefabs you can use to easily add video playback to your project. The following steps will create an application that plays back a fullscreen video:

1. Create a new Unity project
2. Import the AVProVideo package
3. From the AVPro/Prefabs folder in the Project window, drag the FullscreenVideo prefab to your Hierarchy window



4. Create a folder called StreamingAssets in your Project window and copy your video

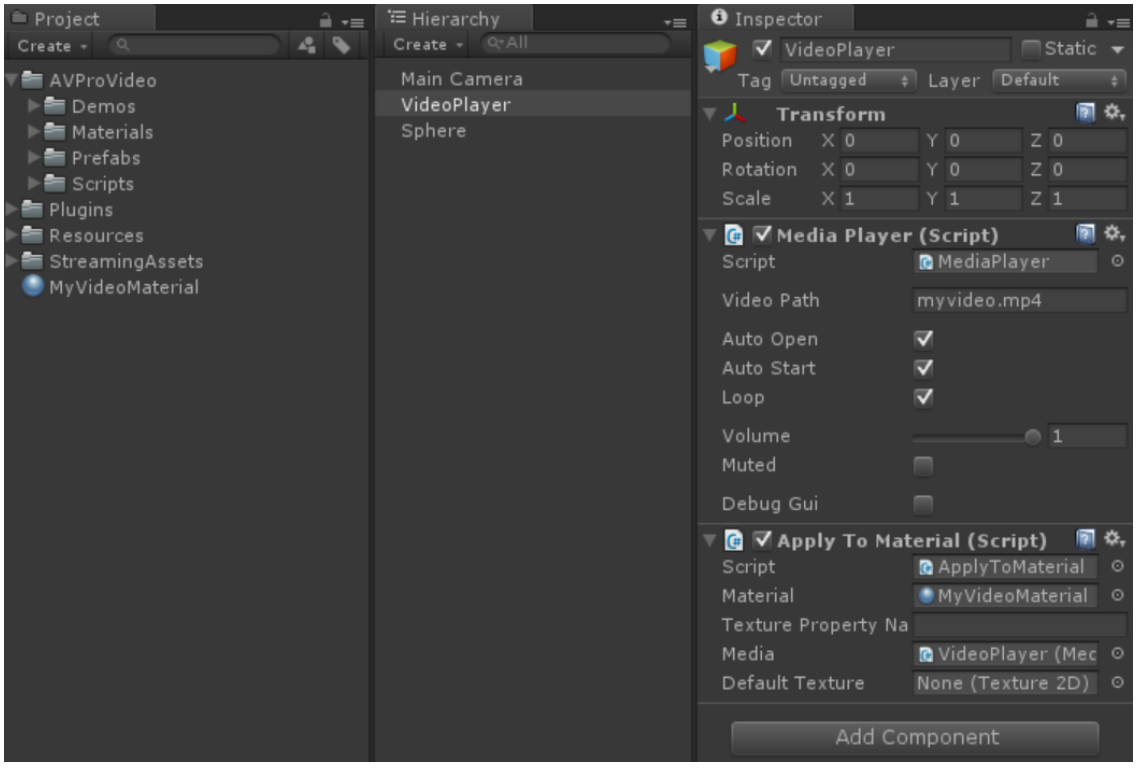
file (say MP4 file) into that folder
5. Enter the file name (including extension) into the Video Path field in the MediaPlayer component (eg myvideo.mp4)
6. Build and deploy your application, the video will be displayed fullscreen

The DisplayIMGUI component script is just one of the components for displaying video. It uses the legacy Unity IMGUI system which always renders on top of everything else. Try using the DisplayBackground or DisplayUGUI components for more control if you don't want your video to be on top.

5.3 Quick Start: 3D Mesh Video Player using Components

AVPro Video includes a number of easy to use script components you can add to your scene. In this example we show how to use the components to play a video onto a material which is applied to a 3D model in the scene.

1. Create a new Unity project
2. Import the AVProVideo package
3. Create a new GameObject from the "GameObject > AVPro Video > Media Player" menu command
4. Click the "Add Component" button and add "AVPro Video > Apply To Mesh"
5. Drag the Media Player script to the "Media" field in the Apply To Mesh script, this tells the Apply to Mesh script which media player to use
6. Create a sphere via the "GameObject > 3D Object > Sphere" menu command
7. Drag the Mesh Renderer component to the "Mesh" field in the Apply To Mesh script, this tells the Apply to Mesh script which mesh to use



8. Create a folder called StreamingAssets in your Project window and copy your video

file (say MP4 file) into that folder

9. Enter the file name (including extension) into the Video Path field in the MediaPlayer component (eg myvideo.mp4)
10. Build and deploy your application, the video will be displayed on your 3D sphere

6. Usage

6.1 Getting Started

The easiest way to get started is to look at the included demos and see what script components have been used. For video playback you need 3 things in your scene:

1. The video file to play:

   Create a “StreamingAssets” folder in your Project window
   Copy your video file (usually MP4 file, but consult the list of supported formats for your platform below) to the StreamingAssets folder

2. A MediaPlayer script to load and play the video:

   Create a GameObject and add the MediaPlayer script to it
   Set the Video Path field to the name of your video file (e.g. myvideo.mp4)

3. A script to display the video:

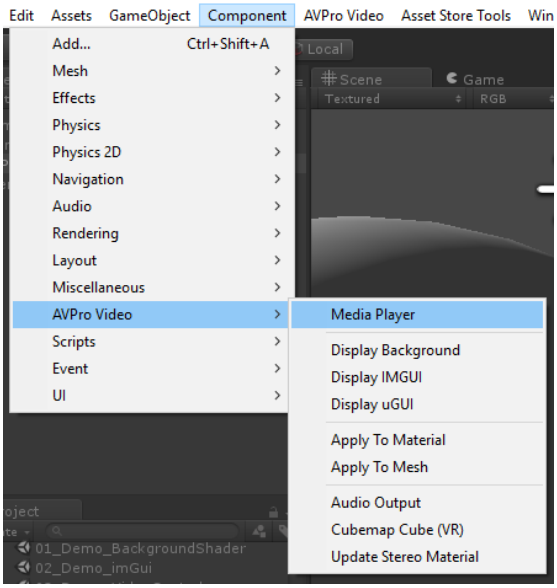
   Decide how and where you want your video file to appear. There are a number of different display component scripts included for different usage scenarios. If you want to display the video on top of everything in your scene just add the DisplayIMGUI script to a GameObject in your scene and set the Media Player field your MediaPlayer component. Other display components work similarly.

6.2 Unsupported Platform Fallback

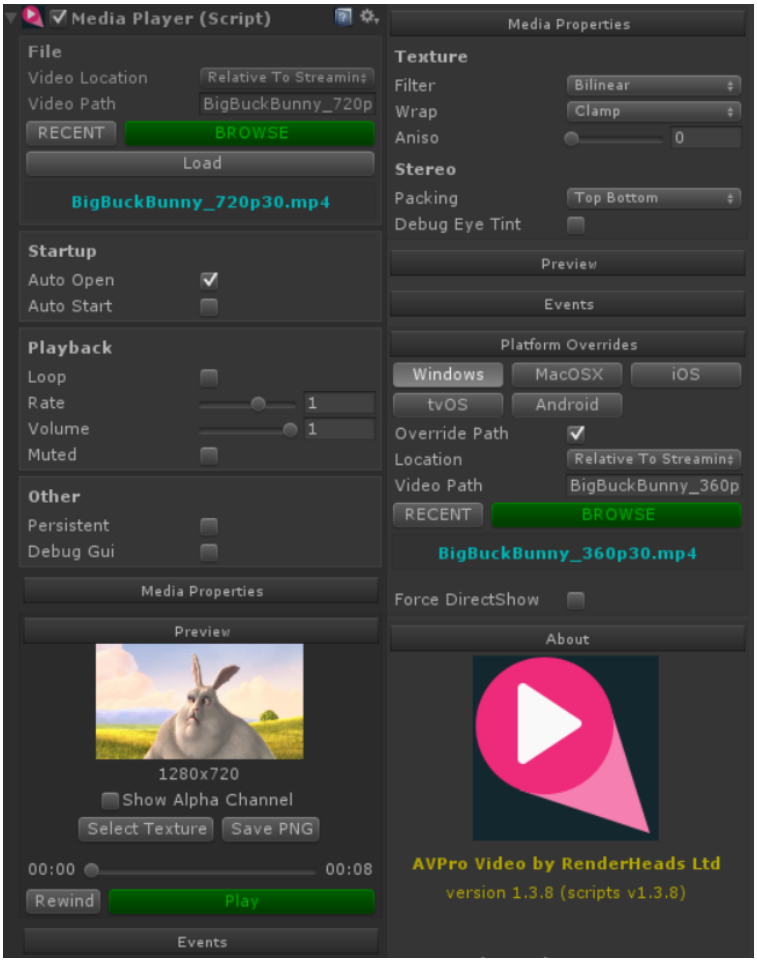
AVPro Video is designed to still function even on platforms that aren’t natively supported. Instead of displaying the actual video though, a dummy 10 second long “AVPro” visual is shown. All of the video controls should still work. For example if you are running your editor in Linux the dummy video player will appear in the editor and the real video will appear when you deploy to supported platforms. If you deploy to an unsupported platform such as Samsung TV you will also see the dummy video player. The code is easily extendable to add custom video players for any unsupported platform.

6.3. Components

Included are a number of components to make this asset easy to use. The components are located in the AVProVideo/Scripts/Components folder or you can add them from the Components menu:



6.3.1 Media Player Component



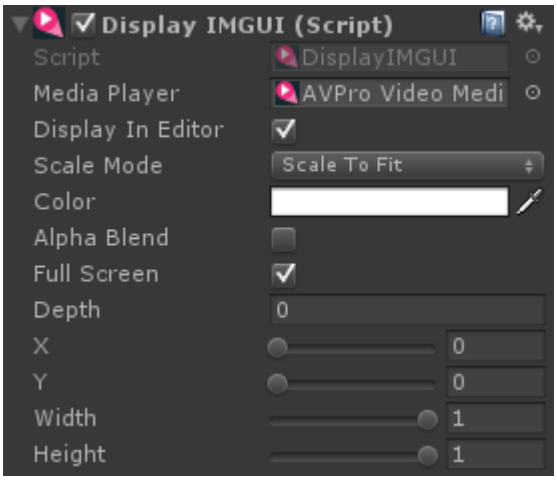
This is the core component for playing media. This component only handles the loading and playback of media and doesn't handle how it is displayed. Use the display script components to control how and where the video is displayed. Fields are:

- Video Location
    - Where to look for the file specified in the Video Path below. This can be an absolute path/URL, or relative to one of the Unity folders. The StreamingAssets folder is the easiest to use. Options are:
        - Absolute or URL
            - This is an absolute path on your device, or an http URL
        - Relative to Project Folder
            - The root folder is the folder above your Assets folder
        - Relative to Streaming Assets Folder
            - The root folder is /Assets/StreamingAssets
        - Relative to Data Folder
            - The root folder is /Assets
            - Unity manual has more information: http://docs.unity3d.com/ScriptReference/Application-dataPath.html
        - Relative to Persistent Data Folder
            - Unity manual has more information: http://docs.unity3d.com/ScriptReference/Application-persistentDataPath.html
- Video Path
    - The file path to the video in the StreamingAssets folder (e.g. myvideo.mp4 or AndroidVideos/myvideo.mp4 if you want to use a subfolder)
- Auto Open
    - Whether to open the file when this component is enabled/starts
- Auto Start
    - Whether to play the video once a video is opened
- Loop
    - Whether to loop the video
- Playback Rate
    - Sets a multiplier that affects video playback speed
    - Please see section 4.13 notes about this
- Volume
    - 0..1 range for audio volume
- Muted
    - Whether the audio is muted
- Persistent
    - Applies DontDestroyOnLoad to the object so that it survives scene/level loads
- Debug Gui
    - Whether to display an overlay with statistics on the video playback - useful for debugging
- Events
    - This event can be hooked up to scripting functions which will get called when a non-looping video completes playback. See the Events section below for more details and a scripting example
- Visual
    - Texture

      Set the desired filtering and wrap mode for the final texture that the frames are written to. Useful if you wish to tile your textures in a specific way.

    - Transparency
        - Set the pack mode used for transparency, where one half of the video contains the colour data, and the other half contains the monochrome alpha channel data
    - Stereo
        - Set the pack mode used for stereo, where one half of the video contains the left eye data, and the other half contains the right eye data
    - Resampler
        - The resampler helps smoothen out jitters in video playback caused by unity and decoder vsync drift. The downsides to using it is that it uses more processing power and GPU memory, and that it is out of sync for videos with audio as it displays the video frames a couple of frames later than the decoder.
        - There are two modes in the resampler:
            - POINT picks the closest frame in the buffer and displays it
            - LINEAR picks the two frames directly before and after the current display time, and displays the interpolated version of the two based on how far away the current time is from both of them
        - Resample buffer size determines how many frames are stored in the buffer. A larger buffer size uses more GPU memory as there are more textures, but increases the chance that the desired frames will be found in the buffer, which will allow for smoother playback. A larger buffer size will also mean a larger delay between the video displayed and the actual video on the decoder.
- Platform Specific
    - These allow you to set a different options per platform, including setting a different file path.
    - Windows
        - DirectShow options
            - Preferred Filters - list the names of filters you want to force, eg “LAV Video Decoder”, “Lav Splitter Source”, “Microsoft DTV-DVD Video Decoder”

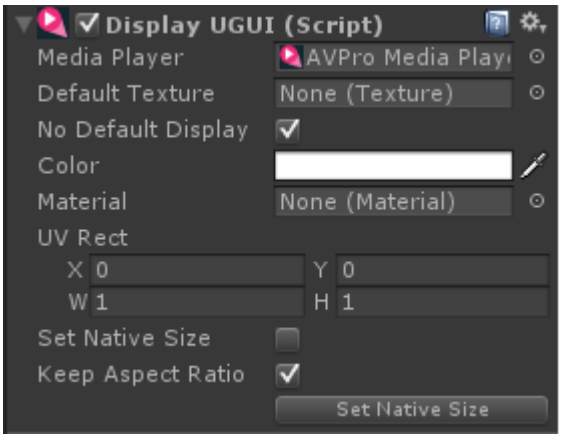
6.3.2 Display IMGUI Component



This is the most basic component for displaying the video. It uses the legacy Unity IMGUI system to display the video to the screen. IMGUI is always rendered on top of everything else in the scene, so if you require your video to be rendered in 3D space or as part of the uGUI system it's better to use the other components. Fields are:

- Media Player
  - The media player to display
- Display in Editor
  - Whether to display the rectangle in the editor - useful for debugging
- Scale Mode
  - How to fit the video to the screen
- Color
  - The color to tint the video, including alpha transparency
- Alpha Blend
  - Whether the video texture controls transparency. Leaving this off for opaque videos is a minor optimisation
- Depth
  - The IMGUI depth to display at. Use this to change the order of rendering with other IMGUI scripts
- Full Screen
  - Whether to ignore the X, Y, Width, Height values and just use the whole screen
- X
  - The normalised (0..1) x position
- Y
  - The normalised (0..1) y position
- Width
  - The normalised (0..1) width
- Height
  - The normalised (0..1) height

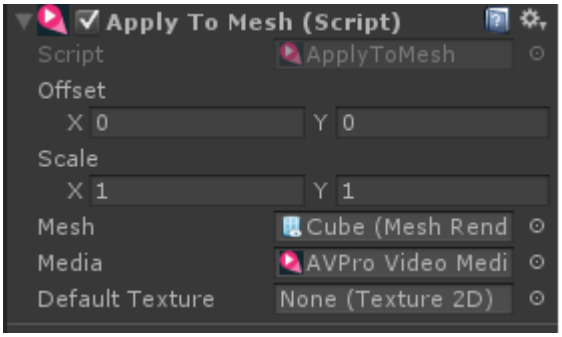
6.3.3 Display uGUI Component



This component is used to display a video using Unity's uGUI system. Field are:

- Media Player
  - The media player to display
- Default Texture (optional)
  - A texture to display while the video isn't playing (while it is buffering for example).
- No Default Display
  - Will not show anything until there are frames available
- Color
  - The color to tint, including alpha transparency
- Material
  - Standard uGUI field, change this to one of the included materials when using packed alpha or stereo videos
- UV Rect
  - Standard uGUI field
- Set Native Size
  - When the video loads will resize the RectTransform to the pixel dimensions of the video
- Keep Aspect Ratio
  - Whether to keep the correct aspect ratio or stretch to fill

6.3.4 Apply To Mesh Component

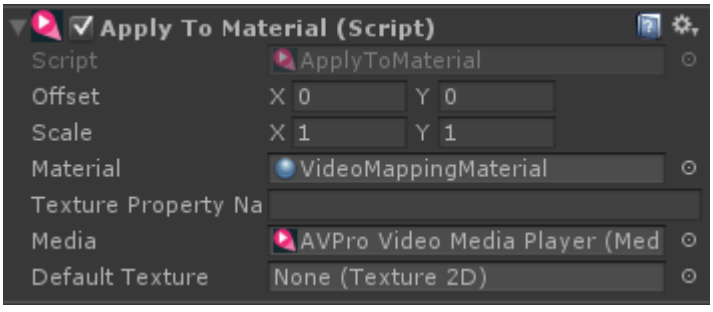


This component takes the texture generated by the Media Player component and assigs it to the texture slot of the material on a 3D Mesh. This is useful for playing videos on 3D

meshes. Field are:

- Offset
  - The XY translation to apply to the texture
- Scale
  - The XY scale to apply to the texture
- Mesh
  - The mesh (renderer) to apply the texture to
- Media
  - The media player
- Default Texture (optional)
  - A texture to display while the video isn't playing (while it is buffering for example).
- Texture Property (optional)
  - By default this script assigns to the main texture (_MainTex) but if you want to assign to another slot you can put the name in here. If you're using the Standard Shader and want the texture to drive the emissive property, set the property name to "_EmissionMap". You may also need to apply a dummy texture to the emissive texture slot in the material to initialise it, and set the emissive colour so it isn't black.

6.3.5 Apply To Material Component



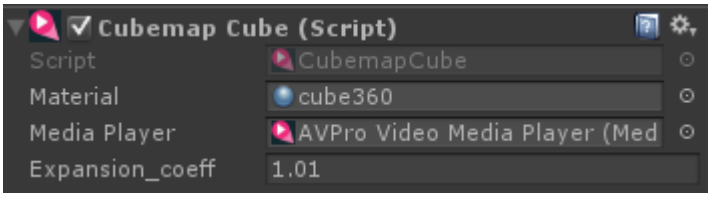
This component takes the texture generated by the Media Player component and assigns it to a texture slot in a Material. This is useful for playing videos on 3D meshes. Fields are:

- Offset
  - The XY translation to apply to the texture
- Scale
  - The XY scale to apply to the texture
- Material
  - The material to apply the video texture to
- Texture Property Name (optional)
  - By default this script assigns to the main texture (_MainTex) but if you want to assign to another slot you can put the name in here. If you're using the Standard Shader and want the texture to drive the emissive property, set the property name to "_EmissionMap". You may also need to apply a dummy texture to the emissive texture slot in the material to initialise it, and set the

emissive colour so it isn't black.

- Media
  - The media player
- Default Texture (optional)
  - A texture to display while the video isn't playing (while it is buffering for example).

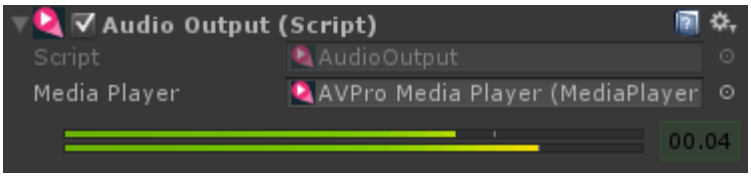
6.3.6 Cubemap Cube Component



This component generates a cube mesh that is suitable for 3:2 cubemap 360 VR videos. Fields are:

- Material
  - The material to apply to the cube. This is usually just a standard unlit material.
- Media Player
  - The media player that will have its video texture displayed on the cube
- Expansion_coeff
  - The value used during enabling to pad the edges to the video to prevent bilinear bleed artifacts. Default is 1.01.

6.3.7 Audio Output Component



This component currently only supports the Windows (UWP, Windows 8 and above), macOS, iOS and tvOS platforms and is used to pass audio from the Media Player to Unity. This allows audio effects, 3D placement and 360 VR spatialisers to be used. Fields are:

- Media Player
  - The media player that will have its audio outputted via Unity

Notes:

- iOS / macOS / tvOS
  - Currently only local files and progressive downloads are supported, not HLS.

6.4 Scripting

6.4.1 Namespace

All scripts in this plugin use the namespace RenderHeads.Media.AVProVideo so be sure to add:

```
using RenderHeads.Media.AVProVideo;
```

to the top of your source files.

6.4.2 Media Player Scripting

Most scripting is likely to center around the MediaPlayer.cs script. This script handles the loading, playback and updating of videos. The script exposes a number of interfaces related to different use cases and can be found in Interfaces.cs

MediaPlayer exposes 3 main interfaces:

- Info Interface
    - The IMediaInfo interface is exposed by the Info property
    - This interface is used to access information about the media, eg:

```
MediaPlayer mp;
mp.Info.GetVideoWidth();
```

- Control Interface
    - The IMediaControl interface is exposed by the Control property
    - This interface is used to control playback, eg:

```
MediaPlayer mp;
mp.Control.Pause();
```

- TextureProducer interface
    - The IMediaProducer interface is exposed by the TextureProducer property
    - This interface is used to get information about how to display the current texture and is used by the Display components, eg:

```
MediaPlayer mp;
Texture videoTexture = mp.TextureProducer.GetTexture();
```

The MediaPlayer script also has a number of methods for controlling loading of media:

- OpenVideoFromFile()

    - Loads the video specified. Useful if you need manual control over when the video is loaded
- CloseVideo()
    - Closes the video, freeing memory

6.4.3 Events

MediaPlayer currently has these events:

- MetaDataReady - Called when the width, height, duration etc data is available
- ReadyToPlay - Called when the video is loaded and ready to play
- Started - Called when the playback starts
- FirstFrameReady - Called when the first frame has been rendered. *(NB: This event currently doesn't get fired for certain browsers in the WebGL build. This includes all non-mozilla non-webkit browsers)*
- FinishedPlaying - Called when a non-looping video has finished playing
- Closing - Called when the media is closing
- Error - Called when an error occurred, usually during loading
- SubtitleChanged - Called when the subtitles change
- Stalled - Called when media is stalled (eg. when lost connection to media stream)
- Unstalled - Called when media is resumed form a stalled state (eg. when lost connection is re-established)

Scripting example:

```
// Add the event listener (can also do this via the editor GUI)
MediaPlayer mp;
mp.Events.AddListener(OnVideoEvent);

// Callback function to handle events
public void OnVideoEvent(MediaPlayer mp, MediaPlayerEvent.EventType et, ErrorCode
errorCode)
{
        switch (et)
        {
                case MediaPlayerEvent.EventType.ReadyToPlay:
                mp.Control.Play();
                break;
                case MediaPlayerEvent.EventType.FirstFrameReady:
                Debug.Log("First frame ready");
                break;
                case MediaPlayerEvent.EventType.FinishedPlaying:
                mp.Control.Rewind();
                break;
        }

        Debug.Log("Event: " + et.ToString());
}
```

6.5 Platform Specific Scripting

6.5.1 Windows Store / UWP / Hololens

See the Demos/Scriptlets/NativeMediaOpen.cs script for details on how to load directly from the native file system (eg from the Camera Roll)

6.6 Third-Party Integration

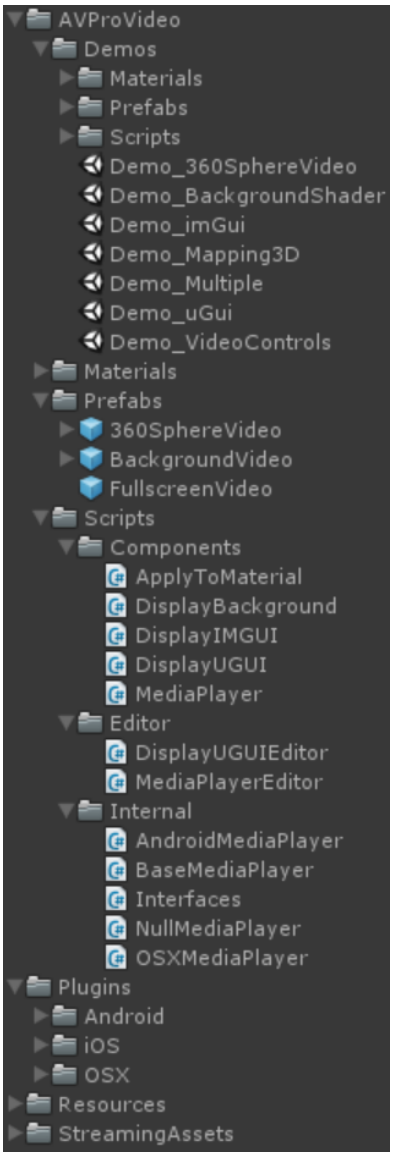
6.6.1 PlayMaker Support

AVPro Video includes over 64 components (actions) for PlayMaker, the popular visual scripting system created by Hutong Games. This makes AVPro Video much easier to use for PlayMaker users. The actions can be found in the /Scripts/Support/PlayMaker folder

6.6.2 NGUI Support

AVPro Video includes basic support for NGUI, the popular UI system from Tasharen Entertainment. The ApplytoTextureWidgetNGUI component is used to set the AVPro Video texture to an NGUI UITexture widget.

7. Asset Files



7.1 Demos

- Demo_360SphereVideo.unity
    - Demo contains a video player that plays a 360 degree video using equirectangular(lat-long) mapping.
    - The video is applied to a sphere, inside of which is the main camera.
    - If the target device has a gyroscope then moving the device around with rotate the camera to view the video from different angles. For platforms without gyroscope the mouse/touch can be used to look around.
    - A special shader and script are used to allow a single camera to render in stereo on a VR headset. Click on the material to set whether it should display the video as monoscopic, stereo top-bottom or stereo left-right.
- Demo_360CubeVideo.unity
    - Same as the sphere demo above, but using a cubemap 3x2 layout source video.
- Demo_BackgroundShader.unity
    - Basic demo that plays a video using the background material which allows the video to appear behind all content.
- Demo_FrameExtract.unity
    - Shows go to read frames out of the video for saving to disk (jpg/png) or accessing pixel data.
- Demo_imGui.unity
    - Basic demo that plays a video and uses the legacy IMGUI display component to draw the video to the screen.
    - Also has an audio clip to show audio-only media playback.
    - Also has 3 different streaming URLs to demonstrate streaming.
    - IMGUI is drawn on top of all other visual components.
- Demo_Mapping3D.unity
    - Demo containing a video player and a 3D scene
    - Some of the 3D models have the video mapped to them via the ApplyToMaterial script
- Demo_Multiple.unity
    - This demo allows you to programmatically multiple load videos and test multiple videos playing at once. Display is via the AVPro Video uGUI component
- Demo_uGUI.unity
    - This demo shows how to display videos within the uGUI system. It uses the DisplayUGUI component in the canvas hierarchy.
    - It also uses a custom shader to overlay text with a video texture.

- Demo_VideoControl.unity
    - This demo shows how to query the video state and control playback

7.2 Prefabs

- 360SphereVideo.prefab
    - Prefab containing a video player and mapping to a sphere. Useful for playback of equirectangular 360 degree videos
- BackgroundVideo.prefab
    - Prefab containing a video player and a quad model with a special background material applied. This material makes the quad get drawn before everything else so it appears in the background.
- FullscreenVideo.prefab
    - Prefab controls a video player and the IMGUI display component for very easy basic video playback creation

7.3 Scripts

- Components
    - ApplyToMaterial.cs
        - Applies the texture produced by the MediaPlayer component to a unity material texture slot
    - ApplyToMesh.cs
        - Applies the texture produced by the MediaPlayer component to a Untiy mesh (via MeshRenderer) by setting the mainTexture field of all its materials
    - AudioChannelMixer.cs
        - Allows up to 8 channels of audio to be volume controlled. This is useful when redirecting audio to a specific channel/speaker.
    - CubemapCube.cs
        - Generates a cube mesh that can be used for displaying a 3:2 cubemap packed video
    - DisplayBackground.cs
        - Displays the texture produced by the MediaPlayer component behind all other content (not compatible with SkyBox)
    - DisplayIMGUI.cs
        - Displays the texture produced by the MediaPlayer component using Unity's legacy IMGUI system
    - DisplayUGUI.cs
        - Displays the texture produced by the MediaPlayer component using Unity's new uGUI system
    - MediaPlayer.cs
        - The main script for loading and controlling an instance of video playback
    - PlaylistMediaPlayer.cs
        - Allows easy playback of multiple videos in sequence, with optional transitions between them. Uses two MediaPlayer components.
    - UpdateStereoMaterial.cs

        - A helper script for VR stereo rendering to update the camera position variable in a spherical material to help work out which eye to render
    - AudioOutput.cs
        - Used to play audio from the media vis Unity's sound system (currently Windows only)
    - ApplyToTextureWidgetNGUI.cs
        - Applies the texture produced by the MediaPlayer component to an NGUI Texture widget texture slot
- Editor
    - DisplayUGUIEditor.cs
        - The editor script that controls how the DisplayUGUI component is rendered in the Inspector
    - MediaPlayerEditor.cs
        - The editor script that controls of the MediaPlayer component is rendered in the Inspector
- Internal
    - AndroidMediaPlayer.cs
        - Android specific media player
    - BaseMediaPlayer.cs
        - Common base class for all platform media players
    - Interfaces.cs
        - Interfaces and events
    - NullMediaPlayer.cs
        - The fallback dummy media player for unsupported platforms
    - OSXMediaPlayer.cs
        - iOS, tvOS and macOS specific media player
    - WebGLMediaPlayer.cs
        - WebGL specific media player
    - WindowsMediaPlayer.cs
        - Windows specific media player

8. Scripting Reference

AVPro Video is designed to be used mainly with the supplied drag and drop component but there are always times when a bit of scripting is needed. The asset includes sample scenes which give some examples of how to use scripting to control video playback, apply video textures to materials etc which are useful to learn from. The full class reference is available online here:
http://www.renderheads.com/content/docs/AVProVideoClassReference/

In this document we have included a simplified version of the highlights.

MediaPlayer class

The MediaPlayer class is the main class for video playback and is where video files are specified and controlled. This class is mainly controlled via the Unity Inspector UI and for scripting through the interface properties it exposes.

Properties

- Events
  - returns the MediaPlayerEvent class
- Info
  - returns the IMediaInfo interface
- Control
  - returns the IMediaControl interface
- TextureProducer
  - returns the IMediaProducer interface

Methods

All of these methods use the interfaces exposed above and are just handy shortcuts

- void OpenVideoFromFile(FileLocation location, string path, bool autoPlay)
  - Opens the video specified
- void CloseVideo()
  - Closes the current video and frees up allocated memory
- void Play()
  - Starts playback of the video
- void Pause()
  - Pauses the video
- void Stop()
  - Pauses the video
- void Rewind(bool pause)
  - Rewinds the video with an option to pause it as well
- Texture2D ExtractFrame(Texture2D target, float timeSeconds, int timeoutMs)
  - Extracts a frame from the specified time of the current video as a readable Texture2D. This can then be used to save out the pixel data. The texture

must be destroyed by the user. The texture can be passed in again via the "target" parameter to reuse it.

IMediaInfo interface

This interface is used to query properties of the video

Methods

- float GetDurationMs();
    - Returns the duration of the video in milliseconds
- int GetVideoWidth();
    - Returns the width of the video in pixels
- int GetVideoHeight();
    - Returns the height of the video in pixels
- float GetVideoFrameRate();
    - Returns the frame rate of the video in frames per second
- float GetVideoDisplayRate();
    - Returns the actual frame rate achieved by the video decoder
- bool HasVideo();
    - Returns whether the media has visual tracks
- bool HasAudio();
    - Returns whether the media has audio tracks
- int GetAudioTrackCount();
    - Returns the number of audio tracks
- string GetPlayerDescription();
    - Returns a string describing the internal playback mechanism

IMediaControl interface

This interface is used to control loading and playback of the video

Methods

- bool OpenVideoFromFile(string path);
    - Starts loading the file from the specified path or URL. Returns false if any error was encountered. This function is asynchronous so the video properties will not be available immediately. This function shouldn't be used, instead use the MediaPlayer OpenVideoFromFile function.
- void CloseVideo();
    - Closes the video and any resources allocated
- void SetLooping(bool looping);
    - Sets whether the playback should loop or not. This can be changed while the video is playing.
- bool CanPlay();
    - Returns whether the video is in a playback state. Sometimes videos can take

a few frames before they are ready to play.
- void Play();
    - Starts playback of the video
- void Pause();
    - Pause the video
- void Stop();
    - Stops the video (essentially the same as Pause)
- bool IsPlaying();
    - Returns whether the video is currently playing
- bool IsPaused();
    - Returns whether the video is currently paused
- bool IsFinished();
    - Returns whether the video has completed playback
- bool IsLooping();
    - Returns whether the video has been set to loop
- bool IsBuffering();
    - Returns whether a streaming video has stopped and is buffering. A buffering video will resume after it has downloaded enough data.
- void Rewind();
    - Sets the current time to the beginning of the video
- void Seek(float timeMs);
    - Sets the current time to a specified value in milliseconds
- void SeekFast(float timeMs);
    - Sets the current time to a specified value in milliseconds but sacrifices accuracy for speed. This is useful if you just want to jump forward/back in a video but you don't care about the accuracy.
- bool IsSeeking();
    - Returns whether the video is currently seeking. During seeking no new frames are produced.
- float GetCurrentTimeMs();
    - Returns the current time (playback position) in milliseconds
- void SetPlaybackRate(float rate);
    - Sets the current playback rate. 1.0f is normal rate. Negative rates aren't supported on all platforms.
- float GetPlaybackRate()
    - Returns the current playback rate
- void MuteAudio(bool mute)
    - Sets the audio mute or not
- void SetVolume(float volume)
    - Sets the volume between 0.0 and 1.0
- float GetVolume()
    - Returns the volume level between 0.0 and 1.0
- int GetCurrentAudioTrack()
    - Returns the index of the currently enabled audio track
- void SetAudioTrack(int index)
    - Sets the index to select the audio track to enable exclusively
- float GetBufferingProgress()
    - Returns a value between 0.0 and 1.0 representing network buffering

- ErrorCode GetLastError()
    - Returns an error code is an error occurred this frame

IMediaProducer interface

Methods

- Texture GetTexture();
    - Returns a Unity Texture object if there is a texture available otherwise null is returned.
- int GetTextureFrameCount();
    - Returns the number of times the texture has been updated by the plugin. This can be useful to know when the texture was updated as the value will increment each time.
- long GetTextureTimeStamp();
    - Returns the presentation time stamp of the current texture in 100-nanosecond units. This is useful for accurate frame syncing.
- bool RequiresVerticalFlip();
    - Some textures are decoded up-side-down and need to be vertically flipped when displayed. This method returns whether the texture needs to be flipped during display.

9. Supported Media Formats

In general the most common format that is supported are MP4 files with H.264 encoding for video and AAC encoding for audio. This format is supported across all platforms though not necessarily all bit-rates and profiles.

Container support:

	Windows Desktop	macOS Desktop	iOS, tvOS	Android
MP4	Yes	Yes	Yes	Yes
MOV	Yes	Yes	Yes	No
AVI	Yes	No	No	No
MKV	Yes in Windows 10	No	No	Yes Android 5.0+
Webm	Yes in Windows 10 - 1607 Anniversary and above	No	No	Yes
ASF/WMV	Yes	No	No	No
MP3	Yes	Yes	Yes	Yes
WAV	Yes	Yes	?	?

Audio Codec support:

	Windows Desktop	macOS Desktop	iOS, tvOS	Android
AAC	Yes	Yes	Yes	Yes
MP3	Yes	Yes	Yes*	Yes
FLAC	Yes in Windows 10	Yes	Yes	Yes
AC3	Yes	Yes	Yes	?
WMA	Yes	No	No	No
MIDI	Yes	Yes	Yes	?

Vorbis	No	No	No	Yes
Opus	Yes in Windows 10 - 1607 Anniversary and above	No	No	Yes, Android 5.0+
ALAC (Apple Lossless)	No	Yes	Yes	No
µLAW	Yes	Yes	Yes	No
ADPCM	Yes	Yes	Yes	No
Linear PCM	Yes	Yes	Yes	Yss

* audio files only

Video Codec support:

	Windows Desktop	**macOS Desktop**	**iOS, tvOS**	**Android**
HEVC / H.265	Yes in Windows 10	Yes in High Sierra and above	Yes in iOS 11 and above	Yes
H.264	Yes****	Yes	Yes	Yes
H.263	Yes	No	?	Yes
MJPEG	Yes	Yes	Yes	No
WMV	Yes	No	No	No
VP8	Yes*	No	No	Yes
VP9	Yes*	No	No	Yes
Hap	Yes***	Yes	No	No
Hap Alpha	Yes***	Yes	No	No
Hap Q	Yes***	Yes	No	No
Hap Q Alpha	Yes***	Yes	No	No
ProRes 422	No	Yes	No	No
ProRes 4444	No	Yes	No	No
DV	Yes	Yes in Yosemite and above	No	No
Lagarith	Yes, with codec	No	No	No

Uncompressed RGBA	Yes	Yes	?	?
Uncompressed YUV	Yes	?	?	?
Uncompressed R10K	No	Yes in Yosemite and above	No	No
Uncompressed V210	?	Yes in Yosemite and above	No	No
Uncompressed 2VUY	?	Yes in Yosemite and above	No	No

For Windows, other codecs can be played if the DirectShow mode is used and 3rd party codecs are installed. We recommend LAV filters for this purpose as this adds support for almost all codecs.

- * Yes, only in Windows 10 and only 4:2:0. Native VP9 support only comes in Yes in Windows 10 1607 Anniversary Update and above, but it may be available before that via Intel GPU drivers. If you use DirectShow and 3rd party filter then 4:4:4 can be supported. Using Media Foundation no audio codecs (Vorbis or Opus) are supported and will cause the video to fail to load if included.
- *** Requires option “Force DirectShow” to be ticked
- **** Older versions of Windows (Vista and XP) do not have support for H.264 decoding
- ***** iOS doesn’t support MP3 audio tracks in a video file, so best to use AAC instead

Cells with “?” are one’s we’re not sure about. We will do more testing and continue to update this table. For more details on which codecs and what type of encoding is best, see the per-platform details below.

9.1 Android

Android supports many media formats. For a complete list check the Android MediaPlayer documentation here: https://developer.android.com/guide/appendix/media-formats.html and the ExoPlayer documentation here: https://google.github.io/ExoPlayer/supported-formats.html

HEVC (H.265) support was officially added in Android 5.0 (Lollipop) but only as a software decoding implementation on older devices.

We have found that using GearVR on Samsung Galaxy S6 and S7 that H.265 codec works best, with a resolution of 3840x1920 at 30fps, or 2048x2048 at 60fps.

A list of media-player related Android chipsets and which formats they support for hardware decoding: http://kodi.wiki/view/Android_hardware

9.2 iOS, tvOS and macOS

9.2.1 iOS

Many media formats are supported by iOS including H.264. iOS 11 adds support for H.265 (HEVC). The iPhone 7 is the first device with the hardware to support H.265.

The iPhone5C is a low-powered mobile device and has a limitation that the videos must be less than 2 megapixels in resolution.

iOS doesn't support MP3 audio tracks in a video file, so best to use AAC instead.

4K videos should be avoided on iOS due to the devices having low memory in general. AVPro Video does however have the YCbCr video decoding option (See Platform Specific panel in MediaPlayer component), which can help play back these videos.

We recommend only using a single MediaPlayer instance and reusing it for all video loads. The memory issue may be helped by encoding your videos with less reference frames but we haven't tested this for sure.

It has proven difficult getting the true video decoding capabilities of iOS devices. Apple's website has information, but we found it to be slightly inaccurate (for example we can decode 4K video on iPhone5s, which apparently can only do 1080p). It seems that if your device has a 64-bit processor then it will be able to decode 4K H.264, but older devices with 32-bit processors will not.

Device specs according to Apple:

Device	MPEG-4	H.264	MJPEG	AAC-LC	Dolby Audio
iPad 4th generation (Nov 2012)	Maximum Resolution: 640x480 at 30fps, Simple profile. Up to 1.5Mbps In MP4 and MOV containers	Maximum Resolution: 1920x1080 at 30fps, High profile up to level 4.1 In MP4 and MOV containers	Maximum Resolution: 1280x720 at 30fps, Up to 35 Mbps In AVI container with PCM audio	160 Kbps 48kHz stereo In MP4 and MOV containers	Unsupported
iPhone 6 (Sept 2014)	Maximum Resolution: 640x480 at 30fps, Simple profile. Up to 1.5Mbps In MP4 and MOV containers	Maximum Resolution: 1920x1080 at 60fps, High profile up to level 4.2 In MP4 and MOV containers	Maximum Resolution: 1280x720 at 30fps, Up to 35 Mbps In AVI container with PCM audio	160 Kbps 48kHz stereo In MP4 and MOV containers	Unsupported
iPad 5th generation (March 2017 iPad 9.7inch iPad Pro	Maximum Resolution: 640x480 at 30fps, Simple profile. Up to 1.5Mbps	Maximum Resolution: 3840x2160 at 30fps, High profile up to level 4.2	Maximum Resolution: 1280x720 at 30fps, Up to 35 Mbps In AVI	160 Kbps 48kHz stereo In M4V, MP4 and MOV containers	Up to 1008Kbps, 48kHz, stereo or multi-channel In MP4 and

iPhone 6S (Sept 2016) **iPhone SE (March 2016)** **iPhone 7 (Sept 2016)**	In MP4 and MOV containers	In MP4 and MOV containers	container with PCM or ulaw stereo audio		MOV containers
iPod 6	Unsupported	Maximum Resolution: 1920x1080 at 30fps ,Main Profile level 4.1 In MP4 and MOV containers	Unsupported	160 Kbps 48kHz stereo In MP4 and MOV containers	Unsupported

iPad 4 supported formats according to Apple are from here:
https://support.apple.com/kb/sp662?locale=en_US
iPad 5 supported formats according to Apple are from here:
https://support.apple.com/kb/SP751?locale=en_US
iPad 9.7inch supported formats according to Apple are from here:
https://www.apple.com/ipad-9.7/specs/
iPad Pro supported formats according to Apple are from here:
https://www.apple.com/ipad-pro/specs/
iPod 6 supported formats according to Apple are here:
https://support.apple.com/kb/SP720?locale=en_US&viewlocale=en_US

9.2.2 macOS

NOTE: We recommend using the OpenGL Core graphics API (in Player Settings), instead of the default Metal API, as we have had reports of video rendering issues on some GPUs when using our plugin with Metal.

Many media formats are supported by macOS including H.264, H.265, ProRes 422 and ProRes 4444.

macOS Yosemite adds support for

- DV
- Uncompressed R10k
- Uncompressed v210
- Uncompressed 2vuy

macOS High Sierra adds support for

- H.265 / HEVC

9.3 Windows

A full list of supported formats can be found here:
https://msdn.microsoft.com/en-us/library/windows/desktop/dd757927(v=vs.85).aspx
https://msdn.microsoft.com/en-us/windows/uwp/audio-video-camera/supported-codecs

H.264 decoder supports up to profile L5.1, but Windows 10 supports above L5.1 profile:
https://msdn.microsoft.com/en-us/library/windows/desktop/dd797815(v=vs.85).aspx

H.265 decoder specs are here:
https://msdn.microsoft.com/en-us/library/windows/desktop/mt218785(v=vs.85).aspx

Windows 10 adds native support for the following formats:

- H.265 / HEVC
- MKV
- FLAC
- HLS Adaptive Streaming
- MPEG-DASH

Windows 10 and UWP HLS and MPEG-DASH features supported:
https://docs.microsoft.com/en-us/windows/uwp/audio-video-camera/hls-tag-support
https://docs.microsoft.com/en-us/windows/uwp/audio-video-camera/dash-profile-support

Windows 10 Fall Update seems to remove native H.265 / HEVC support for some users and requires them to download the (free) HEVC Video Extension. Before update KB4056892 (4 Jan 2018), users also had to open a H.265 video in the Films & TV app after a restart before AVProVideo could play H.265 videos. This update seems to fix that however.

9.4 Windows Phone / UWP

Details on media supported by this platform can be found is platform are here:
https://msdn.microsoft.com/library/windows/apps/ff462087(v=vs.105).aspx
https://msdn.microsoft.com/en-us/windows/uwp/audio-video-camera/supported-codecs

9.5 WebGL

Support for WebGL platform is still varied and depends on the platform and browser support. Some formats such as AVI file container are not supported at all. As with all other platforms, H.264 video in an MP4 container is the most widely supported format.

Adaptive streaming (such as HLS) is still not supported natively by most browsers, but we have seen it working in the Microsoft Edge and Safari browsers.

For best compatibility make sure to force WebGL 1.0 by going to Player Settings > Other Settings > Auto Graphics API and removing WebGL 2.0. Failure to do so can make videos on Chrome not render.

HLS and MPEG-DASH are not natively supported on all browsers. We have added the ability to include 3rd party javascript libraries to handle these (dash.js and hls.js). See the streaming section for how to implement these.

On newer versions of Safari videos are not allows to auto-play unless given permission by the user (in the preferences menu). This doesn't affect videos that have no audio track so this may be a workaround. More details can be found here:
https://webkit.org/blog/7734/auto-play-policy-changes-for-macos/

Some resources about the supported formats:
https://developer.mozilla.org/en-US/docs/Web/HTML/Supported_media_formats
https://en.wikipedia.org/wiki/HTML5_video#Browser_support
http://www.encoding.com/html5/

10. Support

If you have a bug to report, or a feature request, please use our GitHub Issue tracker:

- Issues: https://github.com/RenderHeads/UnityPlugin-AVProVideo/issues

  Please **do not post your project files on GitHub** - instead email unitysupport@renderheads.com if you are sending us a test project.

For general questions and product information please see:

- Unity Forum: http://forum.unity3d.com/threads/released-avpro-video-complete-video-playback-solution.385611/
- Website: http://renderheads.com/product/avpro-video/
- Feedback: https://goo.gl/forms/WB9W9v4vQtez6B1Y2
- Email: unitysupport@renderheads.com

10.1 Bug Reporting

If you are reporting a bug, please post the issue to our GitHub tracker, and include any relevant files and details so that we may remedy the problem as fast as possible.

1. Essential details
    - Error message
        - The exact error message
        - The console/output log if possible
        - If it's an Android build then an "adb logcat" capture
    - Hardware
        - Phone / tablet / device type and OS version
    - Development environment
        - Unity version

    - Development OS version
    - AVPro Video plugin version
- Video details:
    - Resolution
    - Codec
    - Frame rate
    - Better still, include a link to the video file

2. Reproducible Project

    - For the best chance of a fix, please send us a reproducible Unity project. Do not post this to GitHub, but instead email us at unitysupport@renderheads.com with a reference to the GitHub issue number.

11. About RenderHeads Ltd

RenderHeads Ltd is an award-winning creative and technical company that has been designing and building cutting edge technology solutions since its formation in 2006. We specialise in creating interactive audio-visual software for installations at auto shows, museums, shows and expos.

11.1 Services

- Unity plugin development and consulting
- Bespoke software development:
    - High-end technology for events and product launches
    - Interactive installations and educational museums
    - Video walls
    - Virtual reality experiences
    - Augmented reality apps

11.2 Careers at RenderHeads

We're always looking for creative technical people to join our team. RenderHeads is an innovative software company that develops interactive experiences lead by high-end technology and beautiful graphics. We have created installations for Nike, Ford, Nissan, Dell, Intel, PwC, Shell, Cisco, and others. Our work also appears in the National Maritime Museum Greenwich, Museum of Science and Industry Manchester, Sheikh Abdullah Al Salem Cultural Centre Science Museum Kuwait and Guiness Storehouse Ireland.

We create games for museums and brands, bespoke AR, VR and huge video wall experiences for shows, real-time visualisations for events and audio-visual products for developers and the broadcast industry. All of the software we develop is unique, and our clients have high expectations. We work with some of the latest hardware and software technologies, and each project gives us a new opportunity to explore the edge of possibility.

RenderHeads offer a flexible working opportunity with remote or in-office working. We have offices in Cape Town, South Africa and Glasgow, Scotland, and team members working remotely from around the world.

If you're looking for a new opportunity to push the limits and create awesomeness, please read through the requirements below and email us at jobs@renderheads.com. Send us your CV and links to showreel, past projects or code examples.

General Requirements

- You must be able to show a history of making software, either by professional experience or personal projects
- Pragmatic software development approach - we ship software frequently
- Either very strong technical skills, or a mix of creative and technical skills
- Good communication skills - most of our communication is online, via Slack/Skype/Hangouts and email/Google Docs

Positions we have available:

Video Software Developer

- You would be developing in-house camera and video related software, including our AVPro series of Unity plugins, related to cross-platform video playback, streaming, capture and encoding
- Required experience:
  - Strong C++
  - Multi-threaded programming
- Ideal experience:
  - C# and Unity
  - AVFoundation / Media Foundation / DirectShow / libAV / ffmpeg / gstreamer
  - DeckLink / NDI SDK
  - Low-level MP4 / H.264 / H.265 / HEVC / HLS / DASH / RTSP / RTP
  - Shader development
  - Camera / broadcast experience
  - 360 video workflows
  - 360 audio technologies

Interactive Software Developer

- You would be creating software for a wide variety of technically challenging and creative projects, and implementing cool interactive experiences using the latest technologies
- Features of our typical projects:
  - Most development is in Unity
  - Visualisation and effects
  - Educational games
  - VR experience development
  - Integration with cameras, sensors and hardware
  - UI and animation development
- Required experience:
  - C# and Unity
  - SVN / Git
- Ideal experience:

- iOS / Android development
- An interest in UI and UX
- An interest in improving workflow / tools
- Shadertoy and other graphics rendering tech
- Ability to travel (for on-site installations)

Senior Software Developer

- You would oversee the progress of the other developers in your team: making sure everyone is on track, helping to instill good development practices, helping to grow everyone's experience and skills
- You would tackle some of the more difficult programming problems and make sure that the final software is of high quality
- You would also help to scope and plan approaches for new projects, working closely with the other senior members
- Required experience:
  - Many years of software development
  - Many projects/products released
  - Optimisation
  - Unity and C#
- Ideal experience:
  - Graphics programming
  - An interest in improving workflow / tools (Jenkins, CI, Dev ops)

11.3 Our Unity Plugins

Many of the apps and projects we develop require features that Unity doesn't yet provide, so we have created several tools and plugins to extend Unity which are now available on the Unity Asset Store. They all include a **free trial or demo version** that you can download directly from the website here:
http://renderheads.com/product-category/for-developers/

11.3.1 AVPro Video

Powerful cross-platform video playback solution for Unity, featuring support for Windows, macOS, iOS, Android and tvOS.

11.3.2 AVPro Movie Capture

Video capture to MP4 / AVI / MOV files direct from the GPU. Features include 8K captures, lat-long (equirectangular) 360 degree captures, off-line rendering and more. macOS, iOS and Windows supported.

11.3.3 AVPro Live Camera

Exposes high-end webcams, tv cards and video capture boards to Unity via DirectShow. Windows only.

11.3.4 AVPro DeckLink

Integrates DeckLink capture card functionality into Unity, allowing users to send and receive high-definition uncompressed video data to and from these capture cards.

Appendix A - Frequently Asked Questions (FAQ)

1. **Why won't my high-resolution video file play on Windows?**

   The ability to play high resolution videos depends on the version of Windows operating system and which video codecs you have installed.

   By default AVPro Video will try to use the standard Microsoft codecs that come with Windows. These have limitations, for example:

Decoder	Windows 7 and below	Windows 8+
h.264	**1080p***	2160p (4K)
h.265 (HEVC)	**unsupported***	2160p (4K) or 4320p (8K) if your video driver supports

   * If you want to use 3rd party codecs you can install them to improve the range of resolution and format options available. These come in the form of Media Foundation or DirectShow codecs. The LAV Filters are a great example of DirectShow codecs that will allow for higher resolution video decoding.

2. **Does Time.timeScale affect video playback speed?**

   Yes we have BETA support for time.timeScale and time.captureFramerate. Simply enable the option on the Media Player component in the panel labelled "Global Settings".

3. **Does AVPro Video support playing YouTube videos and live streams?**

   Yes and no. If you enter a YouTube URL into AVPro Video it will not be able to play it because this is the URL to the website and not the video. It is possible to gain access to the underlying MP4 file or HLS stream URL which will work with AVPro Video. This may be against the terms and conditions of YouTube though. We have heard that some users have experimented with using the youtube-dl https://rg3.github.io/youtube-dl/ command-line tool to extract the video URL and then use that for playback within AVPro Video.

4. **How can I get smoother video playback for my Windows VR app?**

   This is usually caused by the video decoding taking too much CPU time. Some videos (especially highly compressed ones) require a lot of CPU time to decode each frame.

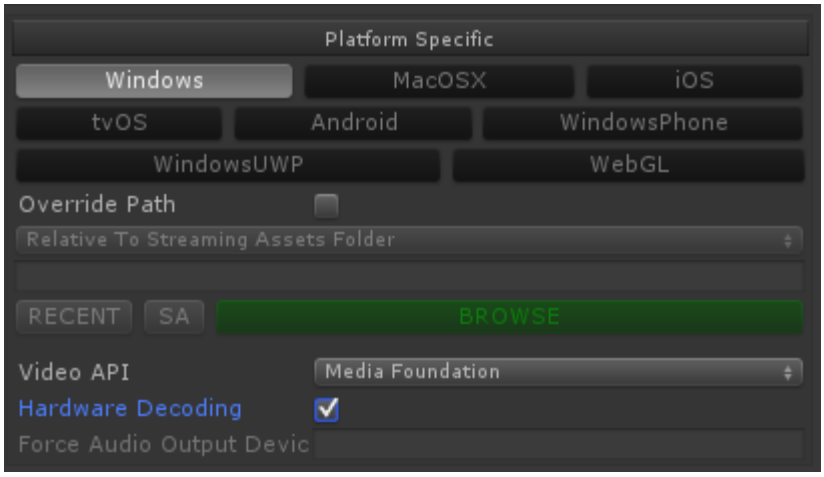
Try enabling the hardware decoding option in “Platform Specific” panel. This will enable your application to use GPU decoding which is much faster. This option is only supported on Windows 8.1 and above and when D3D11 graphics API is used.

You could also try re-encoding your video using settings that are less demanding for the decoder. Most H.264 encoders have a ‘fast decode’ preset, or you can manually change settings such as disabling CABAC, reducing the number of reference frames, reducing bitrate, disabling B-frames, disabling the loop (deblocking) filter etc.

You could also try switching to another video codec that allows for less CPU intensive decoding. Codecs such as H.264 / H.265 are generally very heavy on the CPU. Hap is an extremely fast codec but will result in large files.

5. **Is GPU hardware decoding available?**

Yes it is on most platforms. Android, iOS, tvOS and macOS mostly use GPU decoding - but it depends on the codec. For Windows GPU decoding is enabled by default but is only available for Windows 8.1 and above and when using D3D11 graphics API. You can toggle GPU decoding is enabled via the Platform Specific panel:



6. **Is multi-channel audio supported?**

Audio with more than 2 channels should be supported on desktop platforms. On Windows you will need to set your sound settings in the Control Panel for the number of channels you want to output. If you leave your Windows sound settings as stereo then the audio will get mixed to stereo.

7. **Why isn’t seeking accurate / responsive?**

The way seeking works depends on the platform. Some platforms support frame accurate seeking but most will only seek to the nearest key-frame in the video. In general to improve seek accuracy and responsiveness you can re-encode your videos with more frequent key-frames (smaller GOP). The downside of this is that

the size of the video file will increase (more key-frames = larger file).

We have two seek functions: Seek() and SeekFast(). SeekFast can be used when accuracy isn't important but whether it's faster or not depends on support from the platform and can vary based on the video codec used.

Platform / API	**Accurate Seeking Supported**
Windows Desktop / DirectShow	Yes (but depends on codec used, native Microsoft H.264 decoder doesn't support)
Windows Desktop / Media Foundation	Yes
Windows Store / Phone / UWP	Yes
Android / MediaPlayer	No (but coming in Android "O")
Android / ExoPlayer	Yes
macOS	Yes
iOS / tvOS	Yes
WebGL	Depends on browser

8. **Windows - Why isn't my stereo top-bottom or left-right video displaying correctly?**

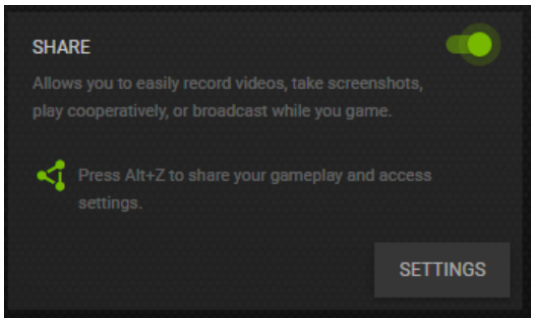
   If your stereo video is only decoding one of the views then this is because there is stereo metadata encoded within the video which causes our Media Foundation decoder to interpret incorrectly. The only way to currently solve this is to remove the metadata from the video. One way to do this is using FFMPEG:

   ```
   ffmpeg -i %1 %1-video.nut
   ffmpeg -i %1-video.nut -crf 10 %1-stripped.mp4
   ```

   We have also had reports that this works:

   ```
   ffmpeg -i in.mp4 -map_metadata -1 -c:v copy -c:a copy out.mp4
   ```

9. **Windows - Why do videos take a long time (1-2 seconds) to open when using DirectShow?**



We have seen this happen when the Nvidia Geforce Experience is installed and the Share option is enabled (ShadowPlay). Disabling the share option or uninstalling this software resolves this issue.

10. **Windows - Why doesn't my H.265 8K video play when I make a build?**

    This is probably because your build is 32-bit. Try making a 64-bit build. See our other notes above for other 8K requirements.

11. **Laptop - Why can't my powerful laptop decode a 8K video?**

    Laptops can have multiple GPUs and will often default to the built-in GPU (eg Intel) instead of the more powerful discrete GPU (eg NVidia / AMD). Try going to the NVidia Control Panel to make the discrete GPU the default and then reload Unity or your application.

12. **Hap Codec - Why don't my Hap encoded videos play?**

    If your platform is Windows desktop then this is usually because the DirectShow video API has not been selected. See the notes above about how to use the Hap codec.

13. **Scrubbing - Why is seeking/scrubbing unresponsive when using a Slider GUI component?**

    This is a common mistake where the OnValueChanged event in the Slider component is set to trigger a method that calls Seek() on the video. The problem is that OnValueChanged doesn't only trigger when the user is interacting with the slider, but also when the video is playing normally and is updating the slider position. So the Update method might be changing the slider position based on the video position, which then causes OnValueChanged to trigger, which causes a seek. You need to guard against this by checking that the user is interacting with the slider, perhaps by adding OnPointerDown and OnPointerUp events. OnPointerDown you would seek to that position and also set a flag. Then in OnValueChanged you would only seek if that flag was set. In OnPointerUp you would unset the flag.

14. **macOS - Errors when using Unity Cloud Build**

    The version of Xcode used by Unity Cloud Build tends to lag behind Apple's releases. Occasionally we will produce a plugin that has been compiled with the most

recent release of Xcode ahead of Unity updating the Cloud Build environment to the latest version, when this happens cloud build will fail with an error similar to the following:

```
ld: could not reparse object file in bitcode bundle: 'Invalid bitcode version (Producer: '1103.0.32.29.0_0' Reader: '1100.0.33.17_0')', using libLTO version 'LLVM version 11.0.0, (clang-1100.0.33.17)' for architecture armv7
```

Unity endeavours to bring the latest version of Xcode to Cloud Build as quickly as possible so if you see this error then we recommend a little patience while Unity catches up.

15. **macOS - Publishing for Mac App Store and get error “Invalid Signature”?**

    We’re not sure why this happens but we’ve had reports that simply deleting all of the .meta files inside the AVProVideo.bundle will resolve this issue.

16. **macOS - Why isn’t my video displaying, or is displaying corrupted video?**

    We have seen this happen when using the Metal graphics API. Try going to Player Settings, unticking the option “Auto Graphics API for Mac” and make “OpenGLCore” at the top of the list.

17. **macOS - Why is the video playback very jumpy when I make a build, but fine in the editor?**

    We have seen this happen when using the Metal graphics API. Try going to Player Settings, unticking the option “Auto Graphics API for Mac” and make “OpenGLCore” at the top of the list.

18. **iOS / tvOS - Why doesn’t my video file play properly?**

    Make sure that your video file doesn’t contain an MP3 audio track. iOS doesn’t support MP3 audio in video files - use AAC for audio instead.

19. **iOS - Can I playback a video from the Videos library?**

    Yes if the video is not content protected and you have it's URL. URL's for items in the video library take the following form:

    ```
    ipod-library://item/item.m4v?id=1234567890123456789
    ```

    You can get the URL of an item by using the MPMediaQuery class from the MediaPlayer framework.

20. **iOS - Can I playback a video from the Photo Library?**

    Yes. You need to use the UIImagePickerController to select the video. This will result in the video being processed into your app's temp folder where you can access

it via the URL returned from the controller.

21. **iOS / tvOS - Why doesn't my HLS m3u8 stream play?**

    First you should check the XCode console for error messages. This can give you a clue about the problem.

    Next you should run your HLS manifest through the Media Stream Validator Tool (https://developer.apple.com/library/archive/technotes/tn2235/_index.html) provided by Apple. iOS and tvOS are VERY strict when it comes to streaming video, so it is possible that a stream will play on macOS, but be refused on iOS or tvOS.

22. **iOS - How can I fix the error message: *"Missing Push Notification Entitlement - Your app includes an API for Apple's Push Notification service, but the aps-environment entitlement is missing from the app's signature"*?**

    In the generated Xcode project:
    - Select the project
    - Select Capabilities
    - Scroll down to Push Notifications
    - And make sure it's turned off

23. **iOS - How can I fix the error message: *"This app attempts to access privacy-sensitive data without a usage description. The app's Info.plist must contain an NSCameraUsageDescription key with a string value explaining to the user how the app uses this data."*?**

    Open the Info.plist file in xcode, and manually add "NSCameraUsageDescription" (Privacy - Camera Usage Description), with a value explaining the reason why you are using the webcam. In our case, this was caused by using the Google Cardboard code and that needs the webcam to read the Cardboard device QR code, so we added the following value "Requires camera for reading Cardboard QR code".

24. **Oculus Go - Why is my Live Steam over LAN not work?**

    It seems that the Oculus go doesn't allow arbitrary ports to be used. Try setting your stream to use port 80.

25. **Android - Why doesn't my huge video file play from StreamingAssets folder?**

    Files in the StreamingAssets folder for Android are packed into a JAR file and so before we can play them we must extract them to the apps persistent data folder. Huge files can take a long time to extract and copy and sometimes they could even cause the device to run out of storage space or memory. For really large files we recommend placing them the videos in other folders on the device and then referencing the absolute path to that file. This also has the added benefit of not having a copy huge files and wait ages when deploying builds to the Android device.

26. **Android - Why does the adb logcat receive so many logs “getCurrentPosition” from MediaPlayer-JNI?**

    Android’s MediaPlayer logs this to verbose channel, simply change your logging filter to ignore verbose logs.

27. **Android - Why doesn’t the video display when using Vuforia?**

    In our test Vuforia doesn’t play well with AVPro Video when multi-threaded rendering is enabled. Simple disable this option in the Player Settings screen.

28. **Android - Why does my build fail with a message about DEX and zip classes?**

    This can happen when there is a conflict in the Java code. We include a JAR called zip_file.jar which contain classes for handling zip files. If you have another plugin in your project that also contains these classes then Unity will fail the build. Read the exact Unity error message and if it is related to zip classes then you can try deleting zip_file.jar file.

    Or you may need to upgrade from JDK 1.7.0 to 1.8.0

29. **Cardboard/Google VR - Why is the image shaking?**

    We’ve had reports that this is caused by enabling the option “track position” in the Gvr head.

30. **Windows 10 - Why doesn’t my H.265 / HEVC video play?**

    Microsoft removed the HEVC codec that was previously included in Windows during the Fall Creators Update in April 2018, Windows 10 version 1709.

    Currently in Windows 10 if you want to play HEVC content you either need to open a HEVC video in the default Film & Movie app, or you need to install this extension:

    “HEVC Video Extensions from Device Manufacturer”
    https://www.microsoft.com/en-us/p/hevc-video-extensions-from-device-manufacturer/9n4wgh0z6vhq

31. **macOS / iOS / tvOS - Why doesn’t my H.265 / HEVC video play?**

    Apple is very strict with the 4 character (FOURCC) ‘tag’ that is used to identify the codec of the video, and only accept videos with the HVC1 tag, whereas many video encoders embed the tag HEV1, which would cause the video not to play on Apple platforms.

    To fix this you either need to be using encoding software that lets you specify the HVC1 tag, or use a tool such as FFMPEG to change the tag for you using this command-line:

```
ffmpeg -i video.mp4 -c:v copy -tag:v hvc1 -c:a copy video-hvc1.mp4
```

32. **Why is my video displaying with the wrong rotation?**

    Videos recorded from mobile devices contain orientation metadata. Some platforms parse this metadata during decoding and produce the texture in the correct final orientation, while others don't and we have to correct it in C# or in the shader. We do handle this for most cases, however if you are using the ApplyToMesh or ApplyToMaterial on macOS, iOS and tvOS then this rotation will not be applied. There are a few ways to fix this:

    1) Re-encode the video with the rotation metadata removed and with the video encoded in its final orientation. Using FFMPEG you can do this via:

    ffmpeg -y -i %1 -metadata:s:v:0 rotate=0 %1-norotation.mp4

    2) Detect the orientation of the video and apply a rotation of the texture coordinates in the shader:

    ```
    Orientation ori = Helper.GetOrientation(_media.Info.GetTextureTransform());
    Matrix4x4 oriMatrix = Helper.GetMatrixForOrientation(ori);
    _material.SetMatrix("_texRotation", oriMatrix);
    ```

    and then the shader define the matrix variable:

    ```
    uniform float4x4 _texRotation;
    ```

    and apply the matrix rotation in the shader shader via something like:

    ```
    o.uv.xy = mul(_texRotation, v.texcoord.xy).xy;
    ```

    You may also have to change the texture wrapping mode in MediaPlayer > Visual > Wrap to Repeat

    If you are using WebGL, then we have found that most browsers (Edge and Chrome) pre-transform the video so that it is in the correct orientation, as there is no mechanism in HTML 5 to retrieve the video orientation/transform. We have found that Firefox doesn't support this though.

33. **iOS - Why do I get this message about privacy permission using camera?**

    Some users have reported getting this message:
    "This app attempts to access privacy-sensitive data without a usage description. The app's Info.plist must contain an NSCameraUsageDescription key with a string value explaining to the user how the app uses this data."

    This seems to be a bug in Unity 5.4.1 and has been resolved in Unity 5.4.2.

34. **Will you support MPEG-DASH or another adaptive streaming system?**

    MPEG-DASH is currently supported on Windows 10, Windows UWP 10 and Android (when using ExoPlayer API) only. We hope to improve support for these sorts of streaming systems eventually. The plugin already supports some basic streaming protocols and for now we're focusing on basic features before we look at advanced features such as these. HLS adaptive streaming is supported on most platforms.

35. **Android - Why is my video playback freezing?**

    We have found that H.264 videos with GOP = 1 (all I-Frames / Key-frames) freeze on Android. GOP interval of 2 seems to be fine.

36. **Windows - Why can't I stream HLS at 4K in Windows 10?**

    Windows 10 (before Anniversary update) seems to limit the stream resolution based on your screen resolution and the monitor DPI scaling setting. To get 4K using HLS you need a 4K monitor and your DPI scaling set to 100%. This issue now seems to be resolved in the latest version of Windows 10.

37. **Windows - Why does the video control react correctly in Windows 7, but not in Windows 8 or above?**

    If you try to call video control function (eg Seek, Mute, SetVolume etc) just after you open the video file, this can cause problems. Video loading can be synchronous or asynchronous, so you need to wait until the video has completed loading before issuing any control commands. In Windows 7 and below the DirectShow API is used by default. Currently the open function works synchronously, so control functions will work immediately after loading. In Windows 8 and above the Media Foundation API is used by default, and this is asynchronous. The best approach is to use the event system to wait for the video to be ready (contain meta data), or to poll the MediaPlayer (Control.HasMetaData()) before issuing control functions.

38. **Why does my HLS / RTSP stream created with FFMPEG work so badly in AVPro Video?**

    Some users have reported issues with creating a stream with FFMPEG where the stream will take a long time to connect and will only send through frames occasionally. We have found that you need to manually add the GOP size parameters to be a multiple of your frame-rate. So for example if your frame-rate is 25 then you should add "-g 25" in your FFMPEG command-line.

39. **Android - How do I authenticate my RTSP stream?**

    RTSP on Android is currently only supported when using the MediaPlayer API. You should be able to pass in your username and password in the URL using the following format:

```
rtsp://<user>:<password>@<host>:<port>/<url-path>
```

Another way is to pass it via HTTP header fields. We also support this. In the MediaPlayer component, go to Platform Specific > Android section and use the JSON HTTP HEADER field to put something like this JSON:

```
{
"SERVER_KEY_TO_PARSE_USER_NAME":"username",
"SERVER_KEY_TO_PARSE_PASSWORD":"password"
}
```

40. **Windows - Why is there a slight colour shift between software and hardware decoding?**

    We're not completely sure why this is. It appears that sometimes that hardware decoder decodes the video slightly darker in places. On the whole the effect is very small, so it only affects videos that contain data that must be reproduced exactly. We have found that the following FFMPEG command does improve the colour matching.

    ```
    ffmpeg -i video.mp4 -y -vf
    "scale=in_range=tv:out_range=tv:in_color_matrix=bt601:out_color_matri
    x=bt709" out.mp4
    ```

41. **Windows Audio - Why does my HLS/DASH stream audio have gaps or play at the wrong rate when using the AudioOutput component?**

    This currently can happen when using adaptive streams as we can't determine the sample rate, and thus cannot set up a resampler to convert to the sample rate Unity is using internally. To solve this you need to set Unity's sample rate in the Audio Settings to match that of the stream (eg 44100).

42. **Audio - Why does my multi-track audio file play all tracks at once?**

    This can happen when all of the audio tracks are set to default. You need to encode the video with only one of the tracks set to default. Using FFMPEG you can do this:

    ```
    ffmpeg -i video_noaudio.mp4 -i audio_esp.m4a -i audio_jap.m4a -i
    audio_eng.m4a -map 0 -map 1 -map 2 -map 3 -disposition:a:1 none
    -disposition:a:2 none -codec copy output.mp4
    ```

43. **Which DRM content protection schemes does the plugin support?**

    See the DRM section in this documentation.

44. **How do I flip my video horizontally?**

    a. In the MediaPlayer component, in the Media Properties section, set the wrap mode to "Repeat".
    b. If you're using the ApplyToMesh component, then you just need to set the

Scale X value to -1

c. If you're using the ApplyToMaterial component, then you just need to set the Scale X value to -1
d. If you're using the DisplayIMGUI component, then this doesn't have any way to flip video without editing the script.
e. If you're using the DisplayUGUI component, then just set the UVRect W value to -1.

45. **Windows - Why doesn't the plugin support Windows XP SP2 and below?**

Modern Visual Studio compilers require XP SP3 as a minimum. One of the reasons for this is builds reference the function GetLogicalProcessorInformation() in Kernel32.dll which isn't present in XP SP 2. The lack of this function causes our DLL to fail to load. More information here:
https://connect.microsoft.com/VisualStudio/feedback/details/811379/

46. **How do I use a video in a reflection probe?**

If you want a video to appear as part of a reflection in your scene, then you can add a reflection probe, set to update in real-time and refresh every frame.

Then add geometry to your scene and use the ApplyToMesh component to apply the video to that geometry. If you are using a equirectangular video then you can apply the video to our OctahedronSphere mesh and use a material on it with the InsideSphere Unlit shader. Then as long as your reflection probe is inside this sphere, it should collect the video.

If you don't want this geometry to appear in your scene, then you can either move the geometry and reflection probe to another area of your scene (you will then have to set the Anchor Override on your MeshRenderer ot make it use that reflection probe), or create a new Layer name and assign it to the geometry and reflection probe, also setting the Culling Mask of the reflection probe so it only renders that geometry. Then in your other cameras you will need to exclude that layer from the camera culling mask list.

47. **How do I use a video as a skybox?**

Take a look at our skybox demo scene

48. **How do I set cookie information?**

Some systems require cookie fields for session/authorisation information. You can set this via the custom HTTP header fields which are specified in JSON format. For example:

```
{
"Cookie":"CloudFront-Policy=xxx;CloudFront-Key-Pair-Id=xxx;CloudFront-Signature=xxx"
}
```

49. **What's the difference between your different Unity video playback plugins?**

We previously had 3 video playback plugins for Unity:

a. AVPro Video
b. AVPro Windows Media (deprecated August 2016)
c. AVPro QuickTime (deprecated May 2016)

Here is a table giving a rough comparison of the plugin features:

	AVPro Video	AVPro Windows Media	AVPro QuickTime
First Released	**2016**	2011	2011
Windows XP-7	**Yes****	Yes	Yes (with QT installed)
Windows 8-10	**Yes**	Yes	Yes (with QT installed)
Windows UWP	**Yes**	No	
Windows Phone	**Yes**	No	
macOS	**Yes**	No	Yes
Android	**Yes**	No	No
iOS	**Yes**	No	No
tvOS	**Yes**	No	No
WebGL	**Yes**	No	
64-bit	**Yes**	Yes	No
2K H.264	**Yes**	Only with 3rd party codec	Yes
4K H.264	**Yes**	Only with 3rd party codec	Yes but very slow
8K H.265	**Yes**	Only with 3rd party codec	No
Streaming URL	**Yes**	Not really	Yes a bit
Hap Codec	**Yes**	Yes	Yes
MP4 Container	**Yes**	Only with 3rd party codec	Yes
Works with VR	**Yes**	Yes best with 3rd party codecs	Not great

ASync Loads	**Yes**	No	Yes
Transparency	**Yes**	Yes	Yes
Speed	**Fast**	Medium, fast with 3rd party codecs	Medium

** Currently only using DirectShow path, adding Media Foundation path soon.

Appendix B - Unity Bugs

Some versions of Unity have new bugs or regressions that prevent AVPro Video from operating as expected.

In our experience it's best to avoid using new major releases of Unity (eg version X.Y.0) and wait for the follow-up releases where bugs introduced in the major update have had time to be fixed.

Please vote and comment on these issues so they get high priority to be fixed.

- **Unity 5.6.0f3 - Present**
  - Has an Android bug that prevents GLSL shaders from compiling, which prevents our OES rendering path to work. This is fixed now: https://issuetracker.unity3d.com/issues/android-ios-glsl-shader-compilation-failure-on-some-mobile-devices
    Latest version where it works it 5.5.4
  - Single-pass stereo mode on the Android GearVR introduces strange shader bugs on some platforms. We added two bug reports, but it won't be fixed as it's a mobile GPU driver issue, so hopefully the drivers got fixed: https://issuetracker.unity3d.com/issues/android-gearvr-left-eye-is-rendered-with-green-distortions-when-using-single-pass-stereo-rendering-method-star-on-mali-gpu
    https://issuetracker.unity3d.com/issues/android-gearvr-both-eyes-are-rendered-in-the-same-color-when-using-single-pass-stereo-rendering-method-star-on-mali-gpu

- **Unity 5.x - Present**
  - Support for Android OBB (split application binary option) seems slightly broken in many versions of Unity. It seems to try to load OBB content from the APK file, so it's as though it's resolving Application.streamingAssetsPath incorrectly. It appears that the first time you deploy to the Android device it works but then subsequent builds will fail. Uninstalling the application from the device and building to a new APK file seems to solve this.

- **Android OES Texture Support**
  - OES Textures used to work reasonably well, but in some versions of Unity they no longer work:
  - https://forum.unity3d.com/threads/android-oes-texture-support.436979/

- **Android Cardboard in Unity 2017.3.1**
  - Unity temporarily broke Cardboard support for non-Daydream devices around these versions:
    2018.1.0 b3 - Fixed in 2018.1.0 b12
    2017.3.1 p1 - Fixed in LTS 2017.4.1 f1
    2017.2.1 p3 - Fixed in 2017.2.2 p3
    2017.1.3 p1 - Fixed in 2017.1.3 p3

- https://forum.unity.com/threads/for-people-targeting-cardboard-on-android-please-update-your-google-vr-sdk-to-1-12-or-later.520986/