RENDERHEADS

AVPro Movie Capture

for macOS, iOS & Windows

The video capture Unity plugin solution for professionals

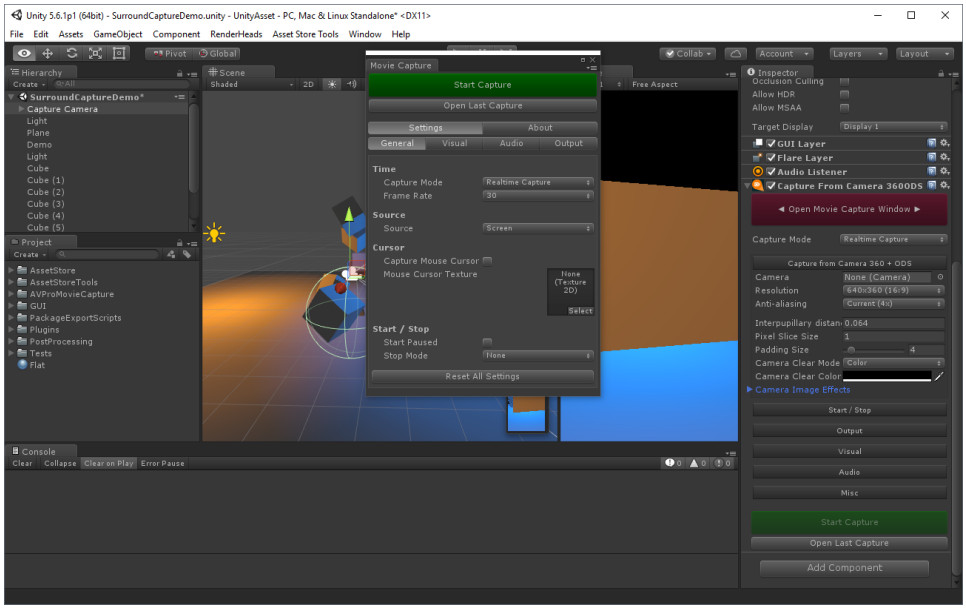
Version 4.4.5

Released 20th October 2020

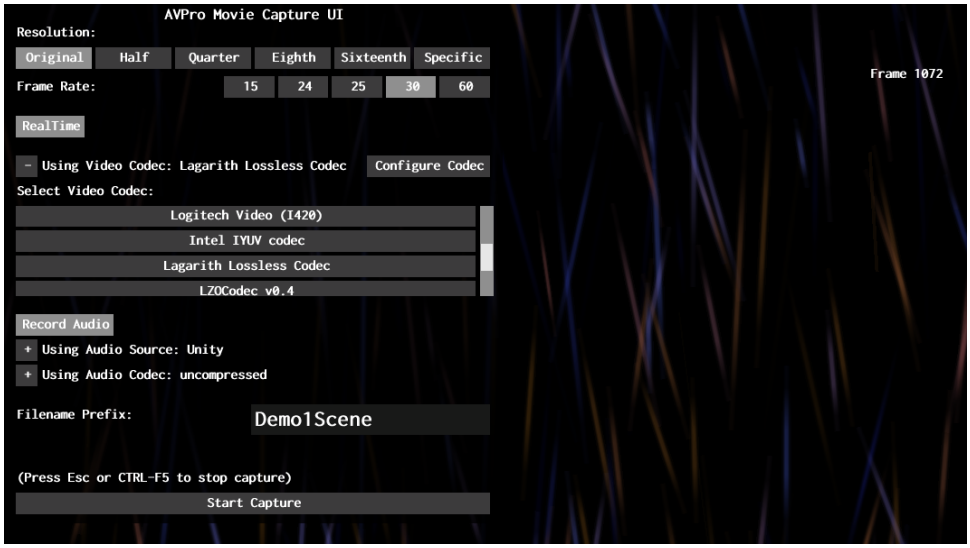
Contents

1. Introduction

AVPro Movie Capture is a plugin for Unity that allows real-time and off-line recording of video and audio, directly to disk as an AVI / MP4 / MOV / PNG / JPG file.

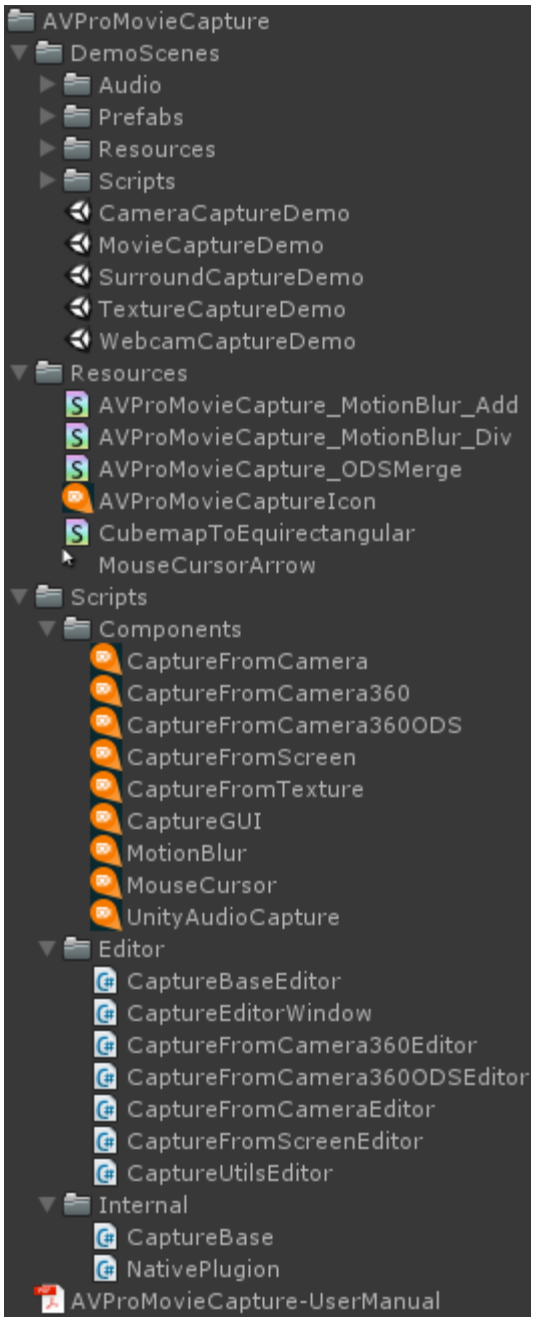


In-editor capturing



In-game capturing

The asset package consists of the following elements:



- Demo Scenes
    - **Demo00-ScreenCapture.unity** - A simple demo showing how to use the CaptureFromScreen component.
    - **Demo01-TextureCapture.unity** - A simple demo showing how to use the CaptureFromTexture component.
    - **Demo02-WebcamCapture.unity** - A simple demo scene showing how to use the CaptureFromWebCamTexture component to capture from WebCamTexture.

    - **Demo03-CameraCapture.unity** - A simple demo showing how to use the CaptureFromCamera component.
    - **Demo04-360Capture.unity** - A simple demo showing how to use the CaptureFromCapture360 component.
    - **Demo05-360CaptureODS.unity** - A simple demo showing how to use the CaptureFromCapture360ODS component.
    - **Demo06-TransparencyCapture.unity** - A simple demo showing how to use the CaptureFromCamera component to capture transparency.
- Plugins
    - **AVProMovieCapture.dll** - The native low-level plugin DLL for 32-bit and 64-bit Windows.
    - **AVProMovieCapture.bundle** - The native low-level plugin for macOS.
    - **libAVProMovieCapture.a** - The native low-level plugin for iOS.
- Component Scripts
    - **CaptureFromScreen.cs** - Drag and drop component to allow easy capturing of the entire scene including IMGUI.
    - **CaptureFromCamera.cs** - Drag ‘n drop component to allow easy capturing from a camera but not IMGUI.
    - **CaptureFromCamera360.cs** - Drag ‘n drop component to allow easy capturing from a camera in 360 degree equirectangular format, without capturing any IMGUI.
    - **CaptureFromTexture.cs** - Drag ‘n drop component to allow easy capturing of a dynamic texture.
    - **CaptureFromWebCamTexture.cs** - Drag ‘n drop component to allow easy capturing of WebCamTexture
    - **CaptureAudioFromAudioListener** - Drag ‘n drop component for capturing Unity audio in real-time mode
    - **CaptureAudioFromAudioRender** - Drag ‘n drop component for capturing Unity audio in off-line mode
    - **CameraSelector.cs** - Allows automatic and dynamic selection of camera during capturing from cameras
    - **CaptureGUI.cs** - Helper component that displays a GUI exposing the capture options of CaptureBase
    - **MotionBlur.cs** - Utility component used to accumulate motion blur.
    - **MouseCursor.cs** - Utility component to draw software cursor for cursor capture
- Editor Scripts
    - Various scripts to handle the drawing of the CaptureFromX components
    - **CaptureEditorWindow.cs** - Created an in-editor Window for capturing directly from Unity without adding any components to your scene.
    - **PostProcessBuild_iOS.cs** - For iOS, enables the UIFileSharingEnabled and LSSupportsOpeningDocumentsInPlace keys in the generated Xcode project’s Info.plist.
- Internal Scripts
    - **NativePlugin.cs** - Wrapper interface to access capture functions in the DLL.
    - **CaptureBase.cs** - Base class

2. Features

- *NEW* Offline audio capture support
- *NEW* D3D12 support
- *NEW* Unity 2020 support
- Very easy to use
- Optimised for high performance
- Real-time capture and offline rendering
- Native support for macOS, iOS and Windows
- Works in the editor and also in standalone builds
- Rendering to 8K resolution
- Many video and image codecs to choose from
- 180 and 360 degree VR capture (mono and stereo)
- Omni-directional stereo (ODS) support for VR renders
- Motion blur rendering
- Linear and Gamma colour-spaces supported
- Alpha channel capture for creating transparent videos
- Records audio directly from Unity or from a system recording device

3. System Requirements

- macOS
    - macOS High Sierra (10.13) and later
    - Metal and OpenGL graphics APIs
    - Unity 5.6 (OpenGL only) * We will be deprecating 5.6 support soon
    - Unity 2017.1 and above
    - Unity 2018.1 and above
    - Unity 2019.1 and above
    - Unity 2020.1 and above
- iOS
    - iOS 11.0 and later
    - Unity 2017.1 and above
    - Unity 2018.1 and above
    - Unity 2019.1 and above
    - Unity 2020.1 and above
    - 64-bit only
    - Metal graphics API only
- Windows
    - Desktop Microsoft Windows platform (32-bit and 64-bit)
    - Windows XP SP3 and higher
    - Unity 5.6 and above
    - Unity 2017.1 and above
    - Unity 2018.1 and above
    - Unity 2019.1 and above

    - Unity 2020.1 and above
    - Codecs for any video formats you want to record to

3.1 Platforms not Supported

- WebGL
- WebPlayer
- Mobile: Android, tvOS, Windows Phone
- Linux
- Windows Store Apps (Note: Windows Metro apps don't support DirectShow which this plugin is built upon. This plugin only supports Windows desktop apps)
- PS4 / XBox / Switch

4. Installation

1. Import the **unitypackage** file into your Unity project.
2. Windows only: Install any 3rd party codecs you want to use if you want to use custom codecs beyond the ones built into Windows.

4.1 Upgrade Notes

If you are upgrading from a previous version, you must make sure Unity doesn't already have the native plugin files loaded, otherwise they will not be updated. The best way to do this is:

1. Close all AVPro Movie Capture windows / tabs in Unity
2. Close all Unity instances
3. Open Unity and your project
4. Immediately import the unitypackage (do not click on the Inspector etc as this can cause AVPro Movie Capture DLL/bundle file to load and become locked, which will prevent upgrade)
5. A Unity editor restart may be required

5. Codecs & Containers

AVPro Movie Capture doesn't include any built-in codecs and instead uses codecs that are already included / registered with the operating system.

5.1 macOS / iOS

5.1.1 H264 Video Codec

The default video codec. Can be used with mp4, m4v and QuickTime container types.

5.1.2 HEVC Video Codec

The high-efficiency video codec, also known as H265. Supported by mp4, m4v and QuickTime container types.
Please ensure your mac has support for hardware HEVC encoding. Results using the software encoder will be... varied.
HEVC with alpha is also supported for capturing transparency, and requires the QuickTime file container to work.

5.1.3 JPEG Video Codec

Compresses video frames using the JPEG codec. Only the m4v and QuickTime container types are supported.

5.1.4 Apple ProRes 422 Video Codec

Apple ProRes 422. Can only use the QuickTime container type. Not supported on iOS.

5.1.5 Apple ProRes 4444 Video Codec

Apple ProRes 4444. Can be used to encode videos with alpha (transparency) channel. Can only use the QuickTime container type. Not supported on iOS.

5.1.6 Uncompressed Video Codec

If you like HUGE media files then this is the codec for you. It's unlikely to be useful. Use the png image sequence capture mode if you want uncompressed video frames. Only supports the QuickTime container type. Uncompressed frames are raw RGB. Not supported on iOS.

5.1.7 AAC Audio Codec

The default audio codec. Works with mp4, m4v and QuickTime container types.

5.1.8 FLAC Audio Codec

The free lossless audio codec. Can only use the QuickTime container type.

5.1.9 Apple Lossless Audio Codec

Apple's lossless audio codec. Can only use the m4v and QuickTime container types.

5.1.10 Linear PCM Audio Codec

16-bit linear PCM interleaved samples at the source sample rate (usually 48000Hz). Can only use the QuickTime container type.

5.1.11 Uncompressed Audio Codec

The raw audio from the audio source, likely to be linear pcm encoded. For instance with Unity this will be 32bit floating point linear pcm at the current audio sample rate (usually 48000Hz). Can only use the QuickTime container type.

5.1.12 PNG Image Codec

Supports an alpha channel.

5.1.13 JPEG Image Codec

Does not support an alpha channel.

5.1.14 TIFF Image Codec

Supports an alpha channel.

5.1.15 HEIF Image Codec

Requires macOS 10.13.4 or later. Supports an alpha channel.

5.1.16 MP4 Container Format

The MPEG-4 container format. This is the default for the H264 and HEVC video codecs and the AAC audio codec.

5.1.17 M4V Container Format

Apple's proprietary video container based on mp4. Supports the H264, HEVC, AppleProRes 422 and AppleProRes 4444 video codecs, and AAC and Apple Lossless audio codecs.

5.1.18 Quicktime Container Format

The Apple Quicktime movie format. Recognised file extensions are: mov and qt. This is the default container for the JPEG, AppleProRes 422, AppleProRes 4444 and uncompressed video codecs and for the FLAC, LinearPCM and uncompressed audio codecs.

5.2 Windows

AVPro Movie Capture supports both the built-in Windows codecs, and any 3rd party codecs installed on the system.

Codec types supported:

- Microsoft Media Foundation
    - The Microsoft HEVC (H.265) video codec is supported
    - The Microsoft H.264 video encoder is supported: https://msdn.microsoft.com/en-us/library/windows/desktop/dd797816(v=vs.85).aspx
- Microsoft DirectShow

    - All DirectShow codecs are supported
- Legacy VFW (Video for Windows)
    - All VFW codecs are supported

There are many codecs out there, each with their own pros and cons. Some codecs are great for real-time encoding, some are lossless and some achieve tiny file sizes using expensive compression, so it is important to consider which codec you use and configure them correctly for your needs.

In general we recommend using Media Foundation codecs for Windows, but if you need more flexibility, DirectShow codecs can be used. Third-party DirectShow codecs we recommend on Windows:

Codec	Real-time	Loss-less	File Size	Alpha Channel
Lagarith	Yes	Yes	Large	Yes*
MagicYUV	Yes	Yes	Large	Yes*
x264	Yes*	Yes*	Small - Medium	No

* after configuring the codec

5.1.1 Media Foundation - H.264 Video Codec

This codec is very useful if you want to quickly create shareable MP4 videos. It requires

Windows 7 or above.

5.1.2 Media Foundation - HEVC (H.265) Video Codec

This codec is very useful if you want to quickly create shareable MP4 videos. It requires Windows 10 or above.

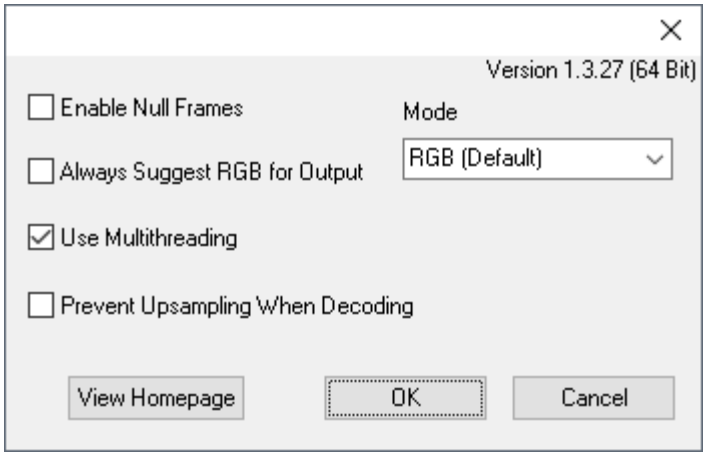
5.1.3 DirectShow - Lagarith Video Codec

Download from: http://lags.leetcode.net/codec.html

Lagarith is a great general purpose codec. It's fast enough for real-time (due to great multi-threading) and lossless. Naturally the files it generates are very large and not suitable for sharing. We use Lagarith as an intermediate codec for real-time capturing and then re-encode to something else like H.264 MP4 offline.

Always check your codec settings. You can do this directly in the plugin via the Configure button, or in 3rd party software like Virtualdub (video > compression menu). Recommended settings:

- Enable "Use Multithreading"
- Set Mode to RGB or RGBA if you need to capture alpha channel



Recommended Lagarith Settings

5.1.4 DirectShow - MagicYUV Video Codec

Download from: http://magicyuv.com/

MagicYUV is another fast codec for real-time capture.

5.1.5 DirectShow - x264 Video Codec

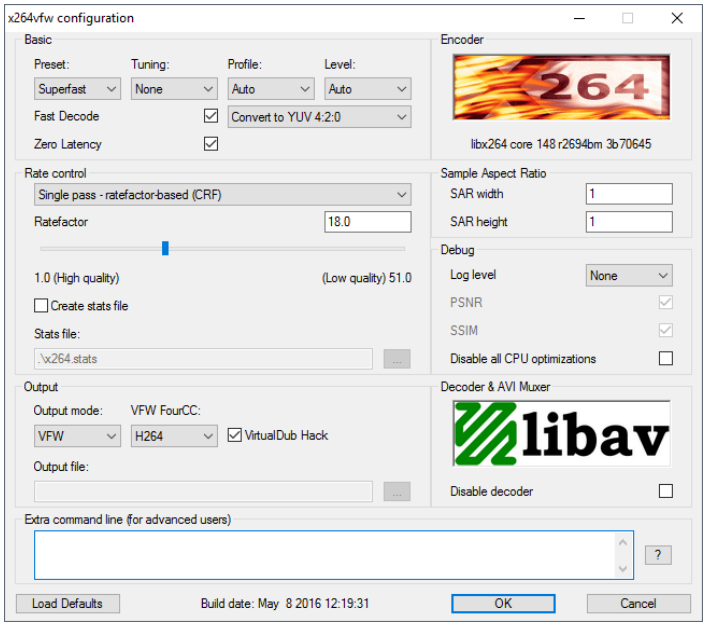
Download from: http://sourceforge.net/projects/x264vfw/

x264 is a highly tunable codec and suits almost any need. By default it's set up for off-line processing which produces tiny files but generally uses way too much CPU for real-time capture. We tweak x264 for real-time capture and use it to directly generate video files suitable for sharing. x264 can also be used with an MP4 muxer to generate MP4 files instead of AVI files (see FAQ).

Always check your codec settings. You can do this directly in the plugin via the Configure button, or in 3rd party software like Virtualdub (video > compression menu). Recommended settings:

- Preset: Fast/Veryfast/Superfast
- Enable "Fast Decode"
- Set "Ratefactor" to your desired quality level
- Enable "VirtualDub Hack"
- Set Debug "Log level" to None
- Checking 'Zero Latency' can help with AV sync and also improve cross-platform playback (for some reason).

Note: You may have to run this configuration twice - once for the 32-bit x264 config, and again for the 64-bit x264 config. These settings are separate for 32bit and 64bit so if you're running 64-bit Unity it will use the 64-bit settings, but then if you build a 32-bit standalone app it will use the 32-bit settings - so it is often best to just build the same "bitness" so you're using in Unity editor to avoid such confusion.

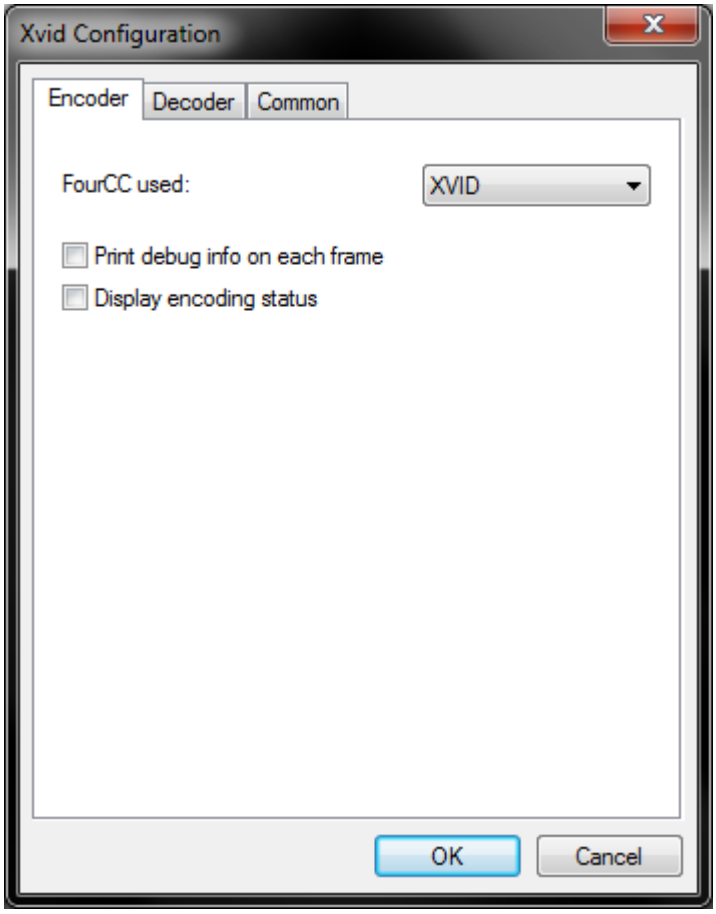


Recommended x264 settings for real-time capture

5.1.6 DirectShow - Xvid Video Codec (Legacy)

Download from: http://www.koepi.info/xvid.html

This codec is a bit old now, but if you want to use Xvid make sure to disable the option "Display encoding status" under "Other options".



Recommended Xvid settings for real-time capture

5.1.7 AVI File Container

AVPro Movie Capture supports the AVI file container by default. AVI is a useful container for capturing but isn't ideal for sharing as it isn't so well supported outside of Windows. AVI also doesn't properly support codecs with B-frames (such as H.264 / H.265). We find AVI to be a useful intermediate container for captures and rendering.

5.1.8 MP4 File Container

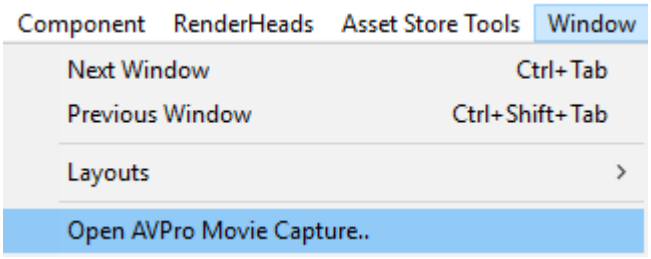
The MP4 container must be the most widely supported file container, so it's perfect for sharing. AVPro Movie Capture does support writing to the MP4 container but there are

currently some limitations on Windows. If you're using Windows 7 or above and you select the Media Foundation H.264 / HEVC codecs, then you can write directly to MP4. If you want to use the x264 DirectShow codec and you want to write to MP4 then you must install a third-party splitter (see FAQ).

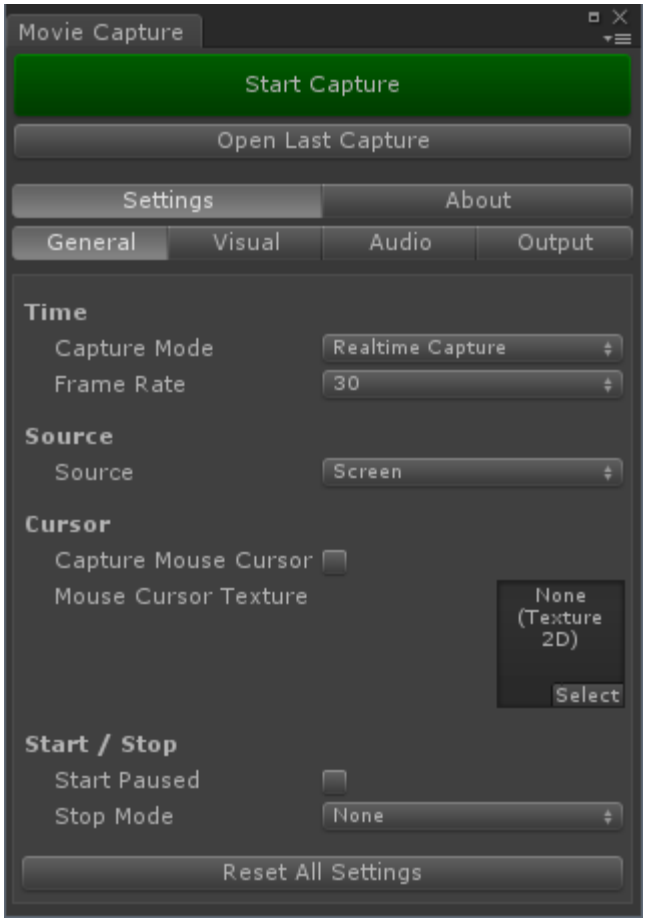
6. Usage

6.1 In-Editor Capture

The editor components allow you to quickly and easily capture videos directly from inside the Unity editor without modifying your scene. Simply open the AVPro Movie Capture editor window:



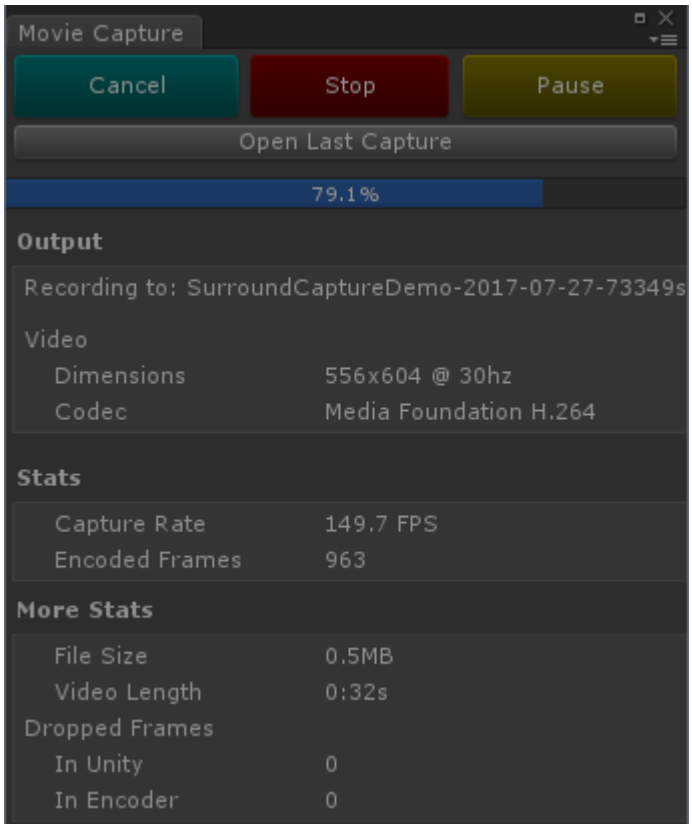
You can now add this UI panel to your editor layout to allow one-click video captures, or open and close it as needed.



In-editor capture window

This panel allows you to configure your recording options and codecs. All settings are

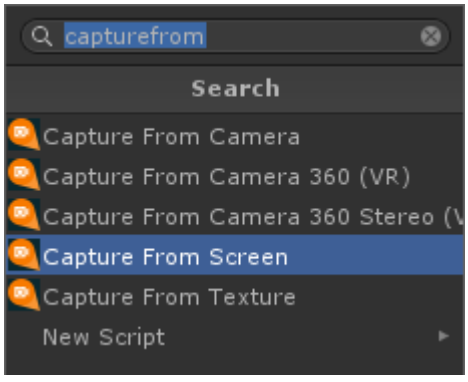
remembered between sessions so once it has been configured once you only need to press the “Start Capture” button. Once a capture starts the statistics of the capture, capture progress and controls are presented.



In-Editor capture session controls

6.2 In-Game Capture

You can add one of the components (CaptureFromScreen, CaptureFromCamera etc) to your scene and trigger them to record directly from your game. This works in the editor and standalone builds.



Components in the “Add Component” button

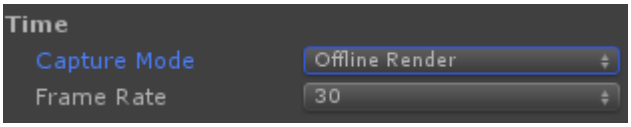
You can read more about these components in the next section.

6.3 Real-time Capturing

For real-time capturing the most important consideration is the codec choice. A codec needs to be selected that is very fast so that it doesn't affect the frame-rate of the game / app being captured.

Lagarith and x264 codecs can be good choices for this and must be configured correctly. The Microsoft H.264 codec can also work well. All codecs are software based, so doing real-time 4K captures will require a powerful CPU. For high resolution captures it may be better to use the offline rendering option.

6.4 Offline Rendering



For visuals that don't require real-time input (cut-scenes, procedural / sequenced animations etc) you have the option to record in offline/non-realtime mode. This mode allows you to capture every frame of animation even if the playback runs very slowly and allows you to capture to any desired frame rate.

For example you may have a mega complex/detailed sequence that renders very slowly in Unity. You can use offline recording to render this to a 60fps video.

Offline rendering isn't suitable for anything that requires real-time input.

Since this type of rendering can take a while, you can use the Auto Stop option to set how many frames / seconds you want it to render for.

6.5 Webcam Recording

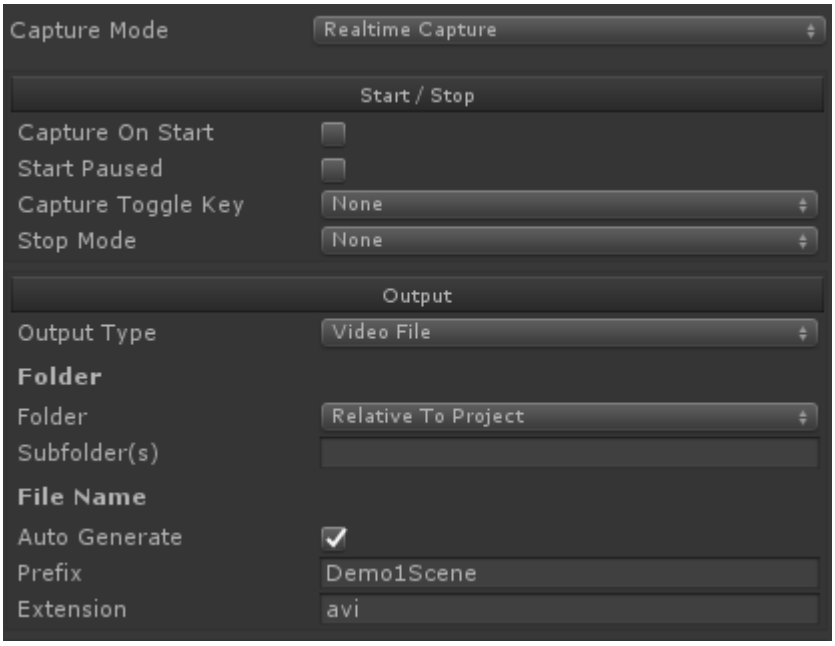
AVPro Movie Capture includes a demo scene that shows how to easily record a Unity webcam to video file. It uses the CaptureFromTexture component as Unity's WebCamTexture is just a normal texture. If you want to capture the audio as well, make sure to select the microphone, or leave the audio device index as -1 to use the default microphone.

On iOS, make sure to fill in the Camera Usage Description field in the Other Settings page of the Player Settings (NSCameraUsageDescription key in the built project's Info.plist), your app will crash otherwise.

7. Components

7.1 CaptureFrom Components

The main components are all prefixed with “CaptureFrom”. They all share common settings:



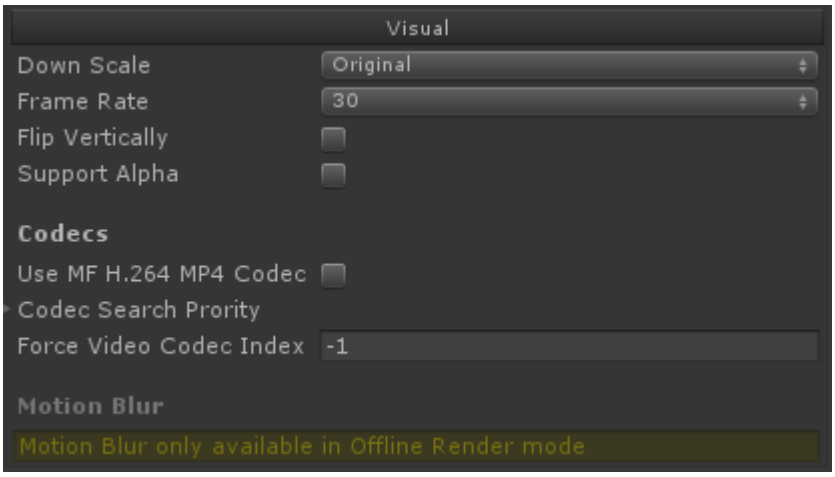
- Capture Mode
    - Selects between real-time capture or offline rendering mode

Start / Stop

- Capture On Start
    - When this GameObject starts, make capture start immediately
- Start Paused
    - When the capture starts, pauses it immediately. This allows the expensive setup of the capture to be done ahead of time so that capturing simply requires a call to Resume() to unpause the capture
- Capture Toggle Key
    - Select a key to toggle the start and end of capturing
- Stop Mode
    - Without a stop mode the capture will continue forever until it is stopped by the user or script
    - You can set a stop mode to make the capture stop when it reaches either a certain number of frames, or a duration in seconds

Output

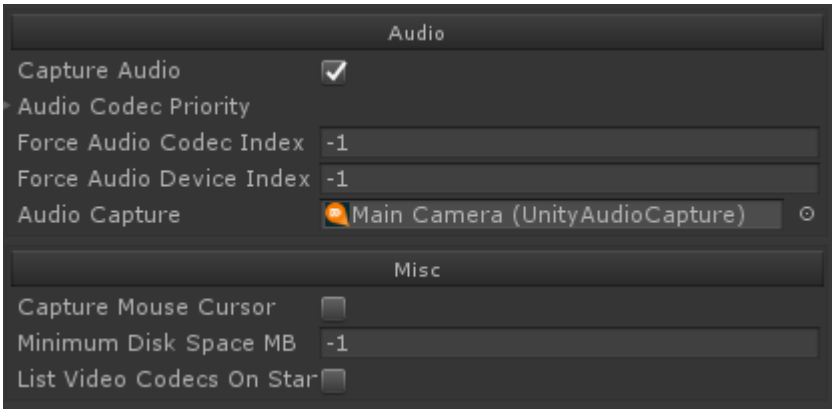
- Output Type
    - Select if you want to output to a video file, or a named pipe (see FAQ)
- Folder
    - Select the folder to output to
- File Name
    - Select the file name to output to, or set it to be auto generated
    - Be sure to set the correct file extension (avi or mp4)



Visual

- Down Scale
    - Reduce the size of the final output
- Frame Rate
    - The frame rate per second for output file
- Flip Vertically
    - Flip the output image vertically (debug only)
- Support Alpha
    - Enable if you want to export transparency. This will require that you use a codec that supports alpha channel. Note that the CaptureFromScreen component doesn't support alpha. Unity doesn't always render correct alpha (it is not a compositing package), so you may need to use custom shaders on your geometry to get the expected results
- Codec Search Order
    - Allows you to list the names of codecs to search for. The first codec in the list that it finds installed on the system will be the one used for encoding. This is useful if you are deploying an app to a system you don't have codec control over so that it can look for a number of codecs before finally falling back to uncompressed. If you want uncompressed simply add a blank entry to the list
- Force Video Codec Index
    - Allows you to override the search priority list and select a codec directly by index
- Motion Blur

- This is an experimental option which is available for offline rendering. It accumulates frames in between the capture frames to create motion blur. This is very useful for high quality rendering but it is very expensive as it is a brute force approach to motion blur.



Audio

- Audio Source
  - Select where audio will be captured from. It's best for performance to only capture audio when you need it
- Unity Audio Component
  - Specify the component to use for capturing audio from Unity.
  - Either a CaptureAudioFromAudioListener (for real-time captures), or CaptureAudioFromAudioRenderer (for offline renders)
  - This is optional - if it is left out then the plugin will create it manually, but this can cause a small delay at the start of capturing
- Audio Codec Priority
  - Allows you to list the names of codecs to search for. The first codec in the list that it finds installed on the system will be the one used for encoding. This is useful if you are deploying an app to a system you don't have codec control over so that it can look for a number of codecs before finally falling back to uncompressed. If you want uncompressed simply add a blank entry to the list
- Force Audio Codec Index
  - Allows you to override the search priority list and select a codec directly by index
- Force Audio Device Index
  - Set to 0 to just use the default microphone recording device, otherwise you can set it to the index to select another device

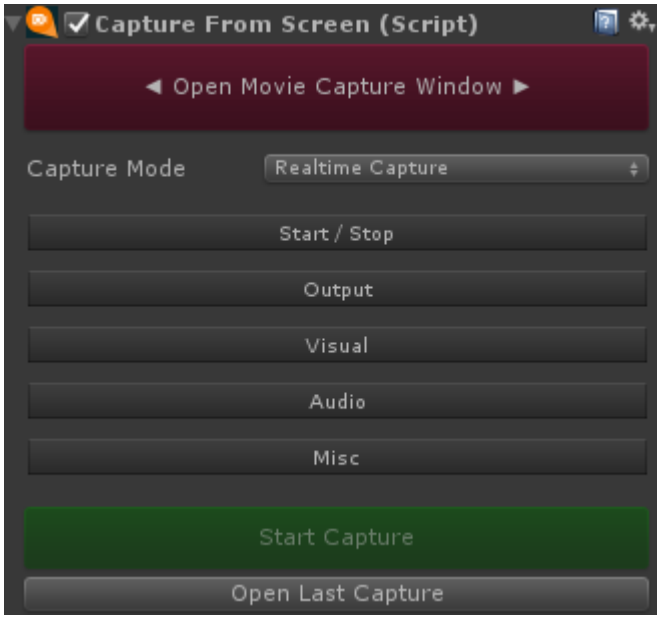
Misc

- Capture Mouse Cursor
  - This option is only available when using the CaptureFromScreen component and allows the mouse cursor to be captured. You have to specify a texture to use for the cursor.

- Minimum Disk Space MB
    - Set to -1 to ignore, otherwise it will keep checking the disk space and stop the capture if the free disk space gets below this number of MB.

7.2 CaptureFromScreen Component

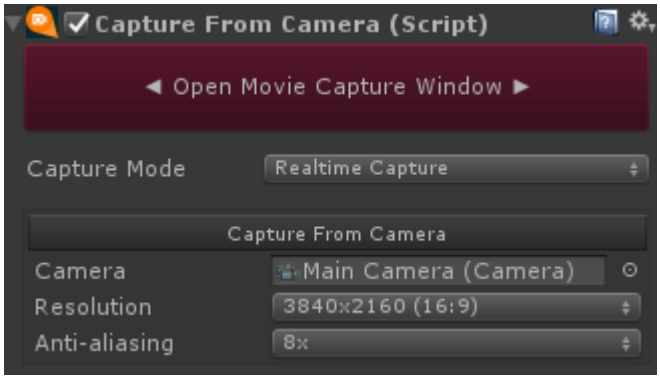
This is the preferred component for capturing as it simply captures everything on the screen (backbuffer), including UI and IMGUI. The hardware mouse cursor is the only thing not captured, but this component has an option to capture the mouse cursor by drawing it using IMGUI.



To use, just add the component to any GameObject in your scene and configure the component.

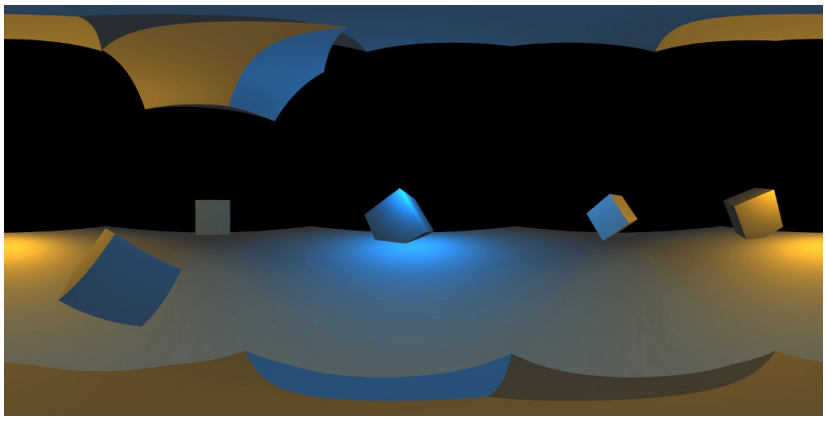
7.3 CaptureFromCamera Component

This component captures the rendering of a Unity camera (including post effects and uGUI). IMGUI is not captured (for this you have to use CaptureFromScreen component). You can capture from a single camera, or from multiple cameras if you have several cameras in your scene that contribute to the final camera rendering. This component allows you to render at a higher resolution than the app is running at, including 8K output (if the codec supports this resolution).

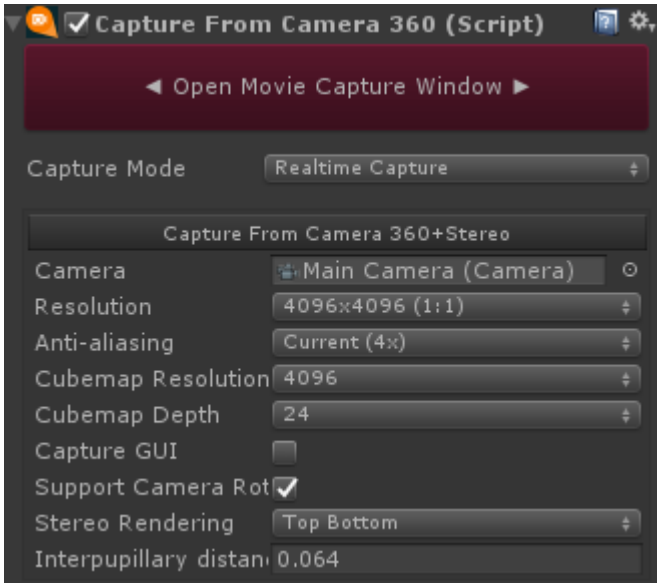


To use, just add this component to any GameObject and assign which Camera it should render. If you want to render from multiple contributing cameras, select the “Use Contributing Cameras” option, and then either manually add the cameras to the list or use the “Find Contributing Cameras” button to fill automatically. The cameras must be listed from first (lowest depth value) to last (highest depth value) rendering order.

7.4 CaptureFromCamera360 Component



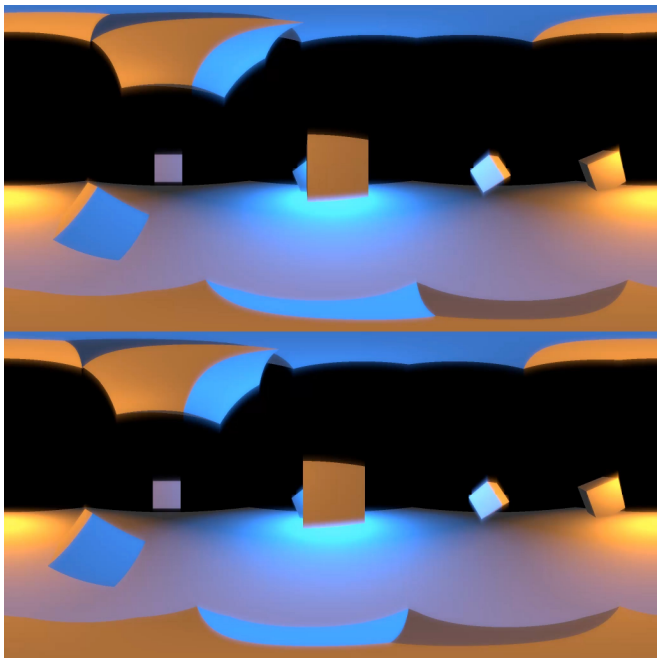
This component allows you to capture full 360 degree views of your scene in equi-rectangular format suitable for viewing in VR.



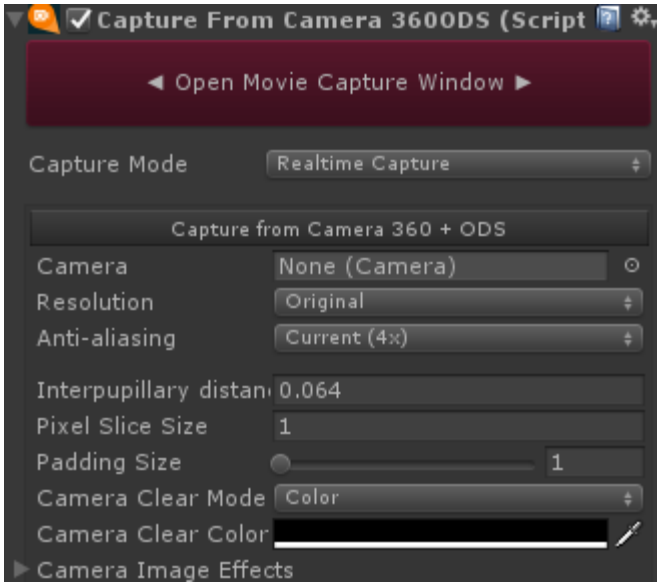
This component works by rendering to 6 faces of a cubemap (so 6 renders of the scene per frame) and then resolving the cubemap into the final equi-rectangular image. Options are:

- Camera
  - The camera to render from
- Resolution
  - The resolution of the final output (before downscale)
- Anti-aliasing
  - Anti-aliasing to use during rendering
- Cubemap Resolution
  - The resolution of the cubemap faces used during rendering
- Cubemap Depth
  - The bit-depth of the cubemap used during rendering
- Capture GUI
  - Whether or not this cubemap renderer needs to capture GUI or not. Capturing GUI requires a slightly slower rendering path
- Support Camera Rotation
  - Whether or not this capture needs to support camera rotation. Doing so requires a slightly slower rendering path, but it is by default enabled
- Stereo Rendering
  - Optionally you can select to create a stereo output in either top-bottom or left-right packing formats. The left eye is always rendered first (top / left). Note that this method of stereo rendering is just a fast approximation. For higher quality stereo the ODS component should be used, but it is much much slower.
- Interpupillary Distance (IPD)
  - The distance between the pupils. This is set to 0.064 which is the standard for an american male adult. You may wish to alter this if creating content for a wide range of users and quality is important. This value is in meters and assumes a scale of 1 unit = 1 meter.

7.5 CaptureFromCamera360ODS Component



This component allows accurate rendering of stereo 360 scenes to equi-rectangular format via a technique called Omni-Directional Stereo (ODS) and based on this paper: https://developers.google.com/vr/jump/rendering-ods-content.pdf



This component takes a very long time to render each frame and is only suitable for offline rendering. This is due to the complex rendering method, but the stereo results are much more accurate than using the stereo option in the CaptureFromCamera360 component. This component always outputs a stereo image and shouldn't be used for monoscopic rendering. Options include:

- Camera
    - The camera to render from
- Resolution
    - The resolution of the final output (before downscale)
- Anti-aliasing
    - Anti-aliasing to use during rendering
- Interpupillary Distance (IPD)
    - The distance between the pupils. This is set to 0.064 which is the standard for an american male adult. You may wish to alter this if creating content for a wide range of users and quality is important. This value is in meters and assumes a scale of 1 unit = 1 meter.
- Pixel Slice Size
    - The size in pixels of each vertical slice.
      Scene renders per frame = (Resolution.width / Pixel Slice)
      Increasing this number will make rendering faster, but will decrease quality. Usually best to leave this set to 1 pixel
- Padding Size
    - The amount of padding in pixels to add to each vertical slice. If you're using post-process effects (bloom etc) then you may need to increase this above 1 so that these effects work.
- Camera Clear Mode
    - How the camera clears
- Camera Clear Color
    - The colour to clear the camera with
- Camera Image Effects
    - Here you can add any Image Effects that are applied to your camera. These effects will then get applied as the rendering is done.

Note that Unity may become unresponsive when using this component. This is because each frame takes a long time to render so the Unity editor will only update at the end of each frame. Hopefully we will improve this in the future to keep the UI responsive. It is best to set the "Auto Stop" mode when using this component so that it will stop at the end of a sequence without user intervention.

Also, please note that this capture component can use a ton of memory if the Unity profiler is enabled due to how many rendering calls it does per frame. Even a simple scene can take over 32GB of memory due to the profiler. You must disable the profiler completely when using this component. This is done by opening the profiler, disabling the record option, closing the profiler, remove the tab if it is docked, then restarting Unity.

7.6 CaptureFromTexture Component

This component captures from any Unity texture (Texture2D, RenderTexture, WebCamTexture etc). The texture to capture is assigned via scripting by calling the SetSourceTexture() method.

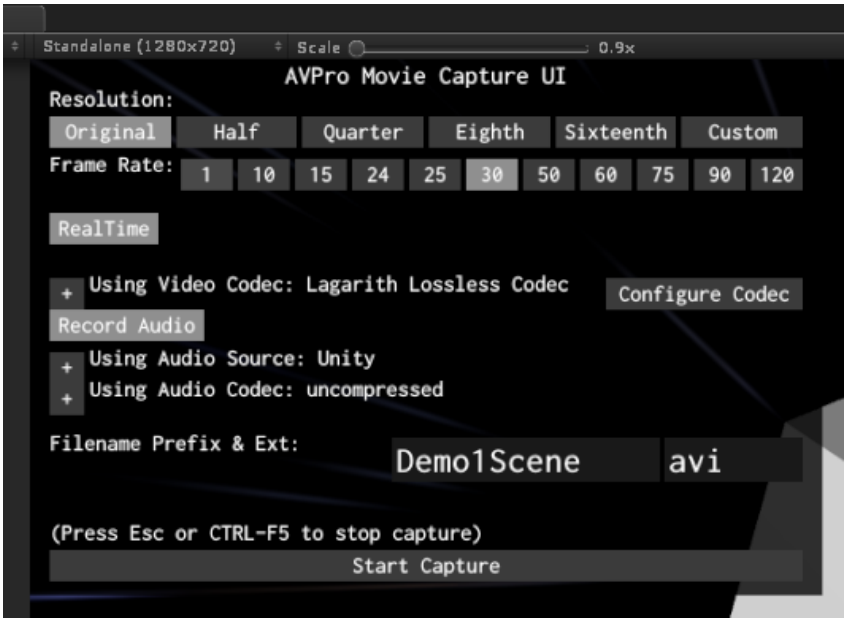
7.7 CaptureAudioFromAudioListener Component

This component can be added to capture the audio from Unity during a real-time capture. This must be added to a GameObject that contains the AudioListener you want to record (usually Main Camera). This component must then be connected to the CaptureFrom* component via the “Unity Audio Capture” field.
One thing to be noted about this component is that it is implemented as a filter, therefore enabling bypass effects on an AudioSource would exclude that AudioSource from being captured.

7.8 CaptureAudioFromAudioRenderer Component

This component can be added to capture the audio from Unity during an offline render. This component must then be connected to the CaptureFrom* component via the “Unity Audio Capture” field.
During capturing this component will set Unity into recording mode, so no audio will be played by Unity.
This component requires Unity 2017.3 or newer.

7.9 CaptureGUI Component



This component renders a GUI to the game view (using IMGUI) to control the CaptureFrom component. It allows the codecs to be selected, capture resolution and frame rate to be chosen and the capture itself to be controlled via Start / Pause / Resume / Stop buttons.
This component is included as it can be useful but it is intended as more of an example than something to include in your builds.

7.10 MouseCursor Component

This utility component can be added when using the CaptureFromScreen component when capturing of the mouse cursor is desired. It renders a software mouse cursor using IMGUI. You may see who cursors on screen - one will be your standard hardware cursor, and the other will be this software cursor. Only the software cursor is captured to video. This component must be assigned to the “Mouse Cursor” field in the CaptureFromScreen component.

7.11 MotionBlur Component

This utility component is added automatically when the motion blur option is selected from the CaptureFrom component. This is not usually added or used manually.

7.12 CameraSelector Component

This component is used by CaptureFromCamera, CameraFromCamera360 and CapturefromCamera360ODS components to give better control over the camera used during capturing. This allows various options to be specified which will determine which camera is used in the scene. This can help for capture situations where the desired capture camera changes over time, or across scene loads.

7.13 TimelineController Component

This component is used for scenes that use Playable Directors / Timeline feature of Unity when doing non-realtime captures. It allows the plugin to take control over the timestep of the Timeline so that it stays in sync with the capture. If you are using the MovieCaptureWindow then this component is automatically used for offline captures. If you are using one of the CaptureFrom components, then simply add this component to the same GameObject and set the TimelineController field in the CaptureFrom component under the Misc section, or via the API using the TimeController property.

8. Scripting

8.1 Basics

All scripts use the namespace: RenderHeads.Media.AVProMovieCapture

All CaptureFrom components are derived from the CaptureBase class. CaptureBase includes most of the functionality, such as controlling when the capture starts and stops. Commonly used methods include:

- bool StartCapture()
    - Starts a capture
- void QueueStartCapture()
    - The same as StartCapture() but should be used when calling from a UI element (eg button)
- bool PauseCapture()
    - Pauses a capture
- bool ResumeCapture()
    - Resumes a paused capture
- bool CancelCapture()
    - Stops a capture immediately. Unlike StopCapture() it doesn't wait for any pending frames to complete compression.
- bool StopCapture()
    - Stops a capture. Note that often (on iOS and macOS, or on Windows when using a Media Foundation codec) the file writing can continue even after the capture has stopped. To determine whether the file is ready for reading/copy/processing you should use the BeginFinalFileWritingAction property to set up an action to be notified when the file writing starts and when it is completed. See the ScreenCaptureDemo.cs script for an example of how to set this up.
- void QueueStopCapture()
    - The same as StopCapture() but should be used when calling from a UI element (eg button)

All settings are publically available in the scripts so they can easily be set in code before starting a capture.

8.2 Custom Usage

If you want to go beyond the component you can access the functionality of the AVProMovieCapture DLL directly or edit CaptureBase.cs to make your changes. You can also derive a new class from CaptureBase to make your own capture component (see the other components to see how this is done).

The DLL has the following functions which are wrapped in NativePlugin.cs:

Global Methods

bool Init();
Global initialisation for the plugin. Returns false if unsuccessful.

void Deinit();
Global de-initialisation for the plugin.

Enumeration Methods

int GetVideoCodecCount();
Returns the number of video codecs on the system.

int GetAudioCodecCount();
Returns the number of audio codecs on the system.

int GetIAudioInputDeviceCount();
Returns the number of audio input devices on the system.

string GetVideoCodecName(int index);
Returns "Invalid" if unsuccessful, otherwise the name of the video codec at index.

string GetAudioCodecName(int index);
Returns "Invalid" if unsuccessful, otherwise the name of the audio codec at index.

string GetAudioInputDeviceName(int index);
Returns "Invalid" if unsuccessful, otherwise the name of the audio input device at index.

Recorder Methods

int CreateRecorderAVI(string filename, uint width, uint height, float frameRate, int format, bool isRealTime, bool isTopDown, int videoCodecIndex, AudioCaptureSource audioCaptureSource, int audioSampleRate, int audioChannelCount, int audioInputDeviceIndex, int audioCodecIndex, VideoHints hints);
Creates a recorder instance for generating video files. An integer is returned which is a unique value specific to this instance of the recorder.

bool Start(int handle);
Starts/resumes recording. Handle is the handle of the recorder instance.

void Pause(int handle);
Pauses recording.

void Stop(int handle);
Stops recording. Handle is the handle of the recorder instance.

void FreeRecorder(int handle);
Releases the instance of the recorder.

Encoding Methods

bool IsNewFrameDue(int handle);
Let's us know whether the encoder is ready for another frame.

void SetTexturePointer(int handle, System.IntPtr texture);
Sets the pointer for the texture to be captured

void EncodeFrame(int handle, System.IntPtr data);
Sends frame to be encoded. "data" points to an array of width x height and with a bit-depth of 32 for RGBA32 videos and 16 for YUY2 videos.

void EncodeAudio(int handle, System.IntPtr data, uint length);
Sends audio to be encoded.

void EncodeFrameWithAudio(int handle, System.IntPtr videoData, System.IntPtr audioData, uint audioLength);
Sends visual and audio frame to be encoded.

Notes

The render thread is also invoked by GL.IssuePluginEvent which is used to grab the currently set texture pointer and send it to the encoder.

For an example of how to use the API, see ScriptCaptureDemo.cs

9. Tips

For best results we recommend:

1. Windows: Use the Media Foundation H.264 video codec where possible. This is a lossy codec but it should be suitable for most use cases.

2. Windows: For lossless capture install and use the Lagarith video codec. It is a free lossless codec with excellent performance. It does produce quite large files though so you may need to convert it to another format before sharing/uploading. You need to configure the Lagarith codec and enable multi-threading and null frames.

3. If you need to convert videos from one codec to another you can use FFMPEG from the command-line, or Handbrake.

4. If a specified codec couldn't be found, a warning is generated and uncompressed video will be produced.

5. For best real-time capture performance create a build and do your captures from the build running in full-screen exclusive mode (set in the Player Settings).

6. Capture a resolution that has a width that is a multiple of 16. This should allow best cache usage and SIMD instructions to be used during memcpy.

7. macOS / iOS: Don't use HEVC unless you are certain that your device has hardware support for encoding.

10. Support

For general questions and product information please see:

- Website: http://renderheads.com/products/avpro-movie-capture/
- Forum: http://forum.unity3d.com/threads/120717-Released-AVPro-Movie-Capture
- Feedback: https://goo.gl/forms/WB9W9v4vQtez6B1Y2

10.1 Bug Reporting / Feature Requesting

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

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

When you are reporting a bug, please include any relevant information and details so that we may remedy the problem as fast as possible. Useful information includes:

1. Unity version
2. Operating system
3. GPU model
4. Rendering API (DX9 / DX11 / OpenGL / Metal)
5. AVPro Movie Capture plugin version
6. Screenshot of the bug
7. Output log
8. Which codec you are using
9. Which plugin component you are using (eg CaptureFromScreen)
10. A copy of the captured video file (if relevant)

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 unique interactive audio-visual software for installations at auto shows, museums, shows and expos.

11.1 Services

- Unity plugin development
- Unity game / interaction / augmented reality development
- Unity consulting

11.2 Jobs 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 the 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/products/

11.3.1 AVPro Video

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

11.3.2 AVPro Movie Capture

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

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 - FAQ (Frequently Asked Questions)

1. **How do I update to the latest version of this plugin?**

   If you have purchased this plugin from the Unity Asset Store then you simply login to the store and check if there is a new version to update to. Make sure you do some from a newly opened Unity and without using the plugin, otherwise the native DLL plugin file can become locked and won't update.

2. **Can this plugin record the audio from Unity?**

   Yes this is supported. If you're capturing in real-time then it will use the CaptureAudioFromAudioListener component, and is supported in all versions of Unity. If you're doing an offline render, then Unity audio capture will use the CaptureAudioFromAudioRenderer component, and this requires Unity 2017.3 or newer.

3. **Where are my movie captures stored?**

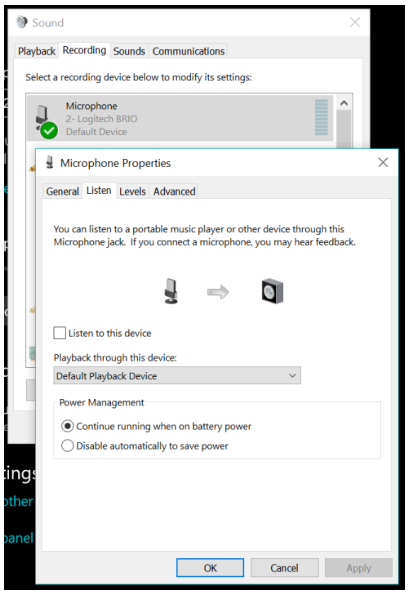
   By default the components auto-generate a filename each time you run a capture. These files are stored in the root of your project (the folder above "Assets"). You can always disable auto-generation of filenames in the component and specify your own file name and location for a capture.

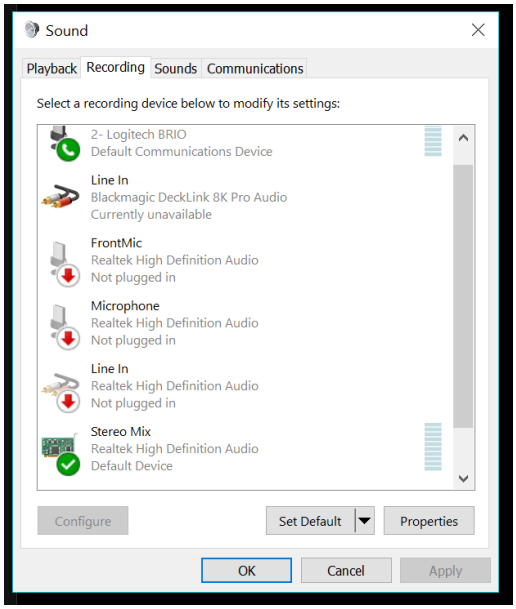
4. **How do I record in-game audio and microphone audio at the same time?**

   Currently this plugin only supports recording audio from a single Windows audio device. There is a "trick" you can use though. In Windows 7 (and perhaps Vista) you may be able to set your microphone to play through the speakers by going to:

   Control Panel -> Sound -> Recording -> Select your microphone -> right click -> Properties -> Listen -> check "Listen to this device".

   You should then hear your microphone recording through your speakers. It's recommended to use headphones during recording to prevent feedback from the speakers into the microphone.





You then need to enable the "Stereo Mix" recording device (it may be hidden by default), and this is the microphone that you will record from in Unity. This virtual microphone will now contain the game audio and also the microphone audio that was set to Listen. This option is also driver specific, as not all sound drivers have a Stereo Mix recording device.

5. **How do I prevent Unity from freezing after doing a recording using the Xvid MPEG4 codec?**

   Open the configuration for the Xvid MPEG4 codec and go to the “other options” page and make sure “Display encoding status” is not selected.

6. **I’m using Autodesk ScaleForm and it’s glitching, how do I get it to record properly?**

   ScaleForm doesn’t seem to like it when render settings (window size, vsync count) are changed while the app/game is running. This plugin removes vsync during recordings which breaks the ScaleForm rendering. Just run your app/game using a quality settings that doesn’t have vsync.

7. **How can I encode to MP4 container file instead of AVI?**

   You can either use the Media Foundation H.264/HEVC codec which is built into Windows 7 and above, or for older systems you can install and use the x264vfw DirectShow codec which requires a MP4 muxer to be installed:

   You need to download the following codec: http://www.gdcl.co.uk/mpeg4/
   Install it from the command-line using “regsvr32 mp4mux.dll” command via an Administrator command-prompt. In the AVPro Movie Capture plugin specify your

target file name ending with ".mp4". Here are some alternative installation instructions:

a. Download the mpeg4.zip file from http://www.gdcl.co.uk/mpeg4/
b. Extract the zip to somewhere on your system (eg C:\Tools\mpeg4)
c. Create a file called register.bat and edit it in notepad and paste the following:

```
regsvr32.exe C:\Tools\mpeg4\x86\mp4mux.dll
regsvr32.exe C:\Tools\mpeg4\x64\mp4mux.dll
pause
```

d. Save the file and close notepad
e. Right click on the register.bat file and choose "Run as administrator"
f. The script should run and register all 4 DLLs to your system

Note that the MP4 container only support certain video and audio codecs (unlike AVI). We recommend using x264vfw video codec and Uncompressed / ADPCM audio codec. You may need to disable audio recording if you aren't using the recommended audio codec.

8. **I have compiled the scripts into a DLL and am now experiencing some unexpected behaviour.**

Some of our scripts have Unity version-specific preprocessor defines which determine how they compile (eg UNITY_5_6). Usually when you build an external DLL these defines are missing and so the incorrect version of the code can be compiled. You need to add the appropriate compiler defines to your build.

9. **My videos aren't capturing correctly, or they appear upside-down**

This can happen when Unity is using OpenGL emulation in the editor (which can happen when your build target is set to other platforms such as Android, iOS etc). Check your Edit > Graphics Emulation settings to make sure no emulation is being used. Check your File > Build Settings and for best results set this to "PC, Mac & Linux Standalone" then press "Switch Platform" and restart Unity.

10. **The plugin crashes when it starts, what could be causing this?**

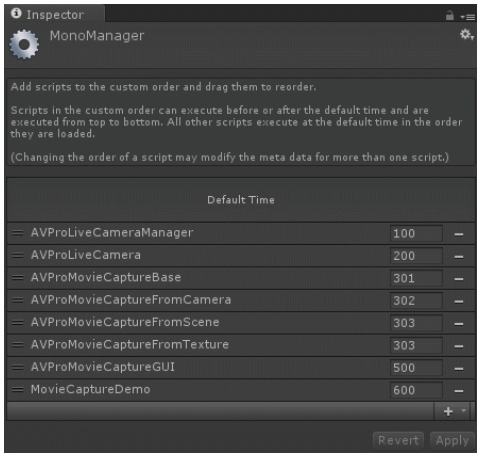
The first thing the plugin does is enumerates all of the video and audio codecs and audio recording devices installed on your system. It is possible that there's a problem with one of these. Try uninstalling all 3rd party codecs you have installed. Once you have it working you can begin reinstalling the codecs again. We have found that the professional version of the Cinepak codec can cause this issue.

11. **Why is my floating point accuracy changing, especially related to epsilon values when using Mathf.Approximately()?**

We've seen rare cases where enumeration of 3rd party codecs changing the FPU state which can affect Unity's Mathf.Approximately() method. We saw this with the x264vfw codec, but it has been fixed in the latest version we tested (July 2017). Otherwise you can try uninstalling other 3rd party codecs.

12. **How can I fix issues when also using the AVPro Live Camera plugin?**

    We've had reports that if you adjust the script execution order then the two plugins will work better together:



13. **How can I create transparent videos with an alpha channel?**

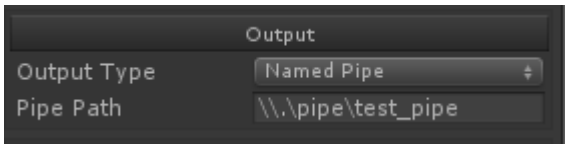
    a. Select a codec that supports alpha channels:
        i. Video Codec:
            1. Uncompressed
            2. Lagarith (Windows only)
                a. make sure it's configured to Mode=RGBA and enable "Use Multithreading".
            3. Cineform (Windows only)
            4. Apple Prores 4444 (macOS only)
            5. HEVC with Alpha (macOS and iOS only)
        ii. Image Codec:
            1. PNG
    b. Use a CaptureFromCamera* component and make sure "Support Alpha" is ticked. Or if you use the Editor Window select "Camera" as the source type and make sure "Support Alpha" is ticked on the Visual tab.
    c. Set the background colour of your capture camera and set the alpha value to zero.
    d. Set the capture camera clear flags to "Solid Color"
    e. If you're using the CaptureFromCamera360ODS component then make sure to set the Camera Clear Color alpha to zero in the capture component settings.

14. How can I include the mouse cursor in my captures?

The Windows mouse cursor is a hardware cursor and is not rendered as part of Unity - so it is not captured by default. You would need to assign a software mouse cursor texture which you can do in Player Settings, or you can use the MouseCursor component included. Only the CaptureFromScreen component supports capture of this mouse cursor.

15. How can I output to a named pipe?

This is an experimental feature that allows the captured video to be written to a named pipe. This allows other software to listen for the incoming video frames. Change the CaptureFrom component Output Type to “Named Pipe”.



Using FFmpeg you can connect to the named pipe:

ffmpeg.exe -y -f rawvideo -codec rawvideo -s 640x480 -r 30 -pix_fmt rgb32 -i \\.\pipe\test_pipe -an -c:v libx264 -pix_fmt yuv420p output.mp4

You must start the capture and then run ffmpeg, otherwise it will not find the named pipe. Currently it only outputs in rgb32 format and you need to specify the resolution and frame rate. Audio isn’t supported.

Another example showing how to stream a video:

ffmpeg -r 30 -vcodec rawvideo -f rawvideo -pix_fmt yuv420p -s 1280x720 -i \\.\pipe\test_pipe -an -f rtp rtp://127.0.0.1:9090

16. **Why is CaptureFromScreen just rendering black in Unity?**

Make sure you don’t have any other “Game” views open, especially ones for other display targets.

17. How do I get my captures off of my iOS device?

In order to use iTunes File sharing, make sure the generated Info.plist file has enabled the UIFileSharingEnabled key. You will then be able to find your app in the file sharing list under the device in iTunes and save your captures out from there. To be able to see your captures in iOS's file app, you will also need to enable the LSSupportsOpeningDocumentsInPlace key.

18. How do I fix seams in my 360 capture caused by particle systems

Unity particle systems will try to face the camera so there will often be seams along the cubemap edges. A few people have solved this by creating a script/shader to align the particles better. More information about this can be found here:
https://youtu.be/1hbcWhifuxE
https://gist.github.com/PatHightree/b2839d03b3b642ad412326907a1c6293

Appendix B - Version History

- **Version 4.4.5 - 20 October 2020**
    - Bug fixes
        - macOS / iOS
            - Fixed issue with offline capture mode where the captured video track length would be shorter than the audio track length
        - Windows
            - Fixed a potential deadlock issue when stopping capture without any frames written

- **Version 4.4.4 - 12 October 2020**
    - Bug fixes
        - Fixed manual audio capture mode not working without a microphone
        - Fixed Android compile issue
        - macOS / iOS
            - Fixed capture failing when using start delay in offline mode

- **Version 4.4.3 - 8 October 2020**
    - Improvements
        - Added a new helper component AudioSourceToWav for capturing Unity audio to WAV files
        - Allowed iOS to run as fast as possible when doing an offline capture
    - Bug fixes
        - macOS / iOS
            - Fixed out of memory issue when software encoder is used
            - Fixed crash when using CaptureFromCamera caused by attempting to use a deallocated texture when performing a second capture
            - Fixed issue when using CaptureFromCamera with capture being solid red when using delayed start
            - Fixed a lock-up when exiting a standalone build by restricting debug log callback to editor only
        - Fixed CaptureAudioFromAudioRenderer to not capture when paused
    - Other
        - D3D12 code now using AVProCore library

- **Version 4.4.2 - 8 September 2020**
    - Improvements
        - StopCapture() no longer blocks while finalising the file to disk when using Windows Media Foundation codecs. Instead it operates as it does on the Apple platforms where it will handle the finalisation in the background, or you can register a listener to check when the file writing is completed via BeginFinalFileWritingAction. The post

process functions also now runs in the background.

- **Version 4.4.1 - 21 August 2020**
  - Improvements
    - Audio from Unity can now be captured in offline renders. This requires Unity 2017.3 or newer. It uses a new component called CaptureAudioFromAudioRenderer
    - Added new CaptureFromWebCamTexture component and updated the Webcam capture demo
    - Removed old #define code related to Unity versions pre-5.6
  - Bug fixes
    - Added a clamp and warning if the selected cube map size is not supported by the system

- **Version 4.4.0 - 30 July 2020**
  - Improvements
    - Windows
      - Added support for native D3D12 capture. This is an experimental new feature.
    - General
      - Added new Manual audio source mode which works in both realtime and offline captures modes allowing audio to be manually sent to the encoder via the API
      - Added new AudioFromAudioClip component which allows an AudioClip to be encoded into the video
      - CaptureFromCamera has better logic for selecting the appropriate texture format to use (HDR, transparent etc)
      - Fixed a bug where an exception would be thrown if none of the codecs listed in search list were available
  - Bug fixes
    - Windows
      - Fixed an issue where AVI uncompressed audio was never selectable

- **Version 4.3.3 - 11 May 2020**
  - Bug fixes
    - macOS / iOS
      - Fixed issue with green component being lost when capturing with transparency using the Metal renderer in the full version

- **Version 4.3.2 - 9 May 2020**
  - Bug fixes
    - macOS / iOS
      - Fixed script compile error introduced in 4.3.1

- **Version 4.3.1 - 4 May 2020**

    - Bug fixes
        - Fixed null exception script regression introduced in 4.3.0 with audio input devices on certain situations

- **Version 4.3.0 - 29 April 2020**
    - Improvements
        - Windows
            - Microphone recording now supported for Media Foundation codecs
        - General
            - Added new DeviceManager class for managing audio input devices (microphones)
    - Bug fixes
        - Fixed some issues where the codec resolver caused some codecs to be unusable

- **Version 4.2.2 - 16 April 2020**
    - Improvements
        - General
            - Some internal code refactoring for the render event functions
    - Bug fixes
        - macOS / iOS
            - Fixed a per-capture memory leak

- **Version 4.2.1 - 15 April 2020**
    - Improvements
        - Windows
            - Added some bitrate and quality encoder hints support for Windows 7
    - Bug fixes
        - Windows
            - Fixed HEVC encoder to allow above 4K encoding
            - Fixed regression that caused Windows 7 H.264 encodings to fail

- **Version 4.2.0 - 9 April 2020**
    - Improvements
        - Windows
            - Added support for HEVC (H.265) encoding on Windows 10
            - Added support for H.264 / HEVC hardware encoding
        - macOS / iOS
            - Added support for writing HEVC with alpha
        - macOS
            - Changed native plugin from dynamic library to signed bundle to resolve macOS quarantine issues
        - iOS

                - Added support for recording audio from a microphone
                - Changed native plugin from static library to framework to prevent symbol collisions with other plugins
            - General
                - Added support for encoder hints which allow specifying of desired bitrates, quality and other settings
                - Added support for capturing from the Scene View window (selectable from the Camera Selector)
                - A lot of script refactoring, many API improvements, including codec system revamp with new classes like CodecManager
        - Bug fixes
            - General
                - Fixed bug where plugin would de-initialise too early when multiple simultaneous captures were still running during application quit / editor, resulting in corrupted video files
                - Fixed issue where Timeline markers/signals were not triggered when making an offline render

- **Version 4.1.0 - 18 March 2020**
    - Improvements
        - Added support for fractional frame rates, eg 29.97 and 59.94 common in the broadcast industry (for off-line capture this requires Unity 2019.2 and above as it uses the new Time.captureDeltaTime field)
        - Better multi-platform support, added platform specific support for certain options, eg image codec and related UI improvements
        - Demo scenes can now upgrade to HDRP/URP automatically as they now use an explicit material instead of the default built-in one
        - Windows
            - MF H.264 encoder now has better default bitrate settings for improved quality
            - MF AAC encoder now supports 1/2/4/6 channel audio instead of previously mono only
    - Bug fixes
        - Fixed cross-platform compile issues when editor build target set to other platforms
        - macOS / iOS
            - Fixed stop mode by encoded frames not working
        - Windows
            - Fixed issue where recording Unity audio with H.264 codec could result in video/audio at the wrong speed, or with audio breakups
            - Fixed issue where first capture with high resolution may not encode due to async texture buffers being full and never emptied
    - Changes
        - OpenGL and Unity 5.x support on macOS is now marked as

deprecated

- **Version 4.0.5 - 21 February 2020**
  - Bug fixes
    - iOS / macOS
      - Fixed corrupt video when capturing from screen with MSAA

- **Version 4.0.4 - 19 February 2020**
  - Bug fixes
    - iOS
      - Fixed linear/gamma issue on iPhone where capture was too dark
    - macOS
      - Fixed PNG alpha issue when capturing from screen
    - macOS / iOS
      - Fixed a crash introduced in 4.0.3 when trying to capture consecutively, and also capture sessions not being free'd when there was no FileWritingHandler listener freeing the sessions

- **Version 4.0.3 - 17 February 2020**
  - Improvements
    - CaptureFromTexture and CaptureFromCamera now support HDR
    - Added new CameraSelector component which allows between control over the camera used for capturing and enables seamless camera capture of between scene loads
    - Added new TimelineController component for keeping Playables timelines in sync with capture
    - Added a new optional delay for the start of capture
    - macOS / iOS
      - Added new class FileWritingHandler which allows the user to check that the stopped capture has completed writing the file. See the ScreenCaptureDemo.cs script for an example of how to use this.
  - Bug fixes
    - Fixed bug in CaptureFromCamera where mixing MSAA/HDR cameras would cause some cameras to not render
    - Fixed bug in CaptureFromTexture where using Linear textures caused capture to be too dark
    - Fixed constant file size query after capture completed
    - Fixed expected plugin version number
    - Fixed some compile errors when targeting other build platforms
    - iOS
      - Fixed function name collision bug with Vuforia plugin

- **Version 4.0.2 - 19 November 2019**
  - Bug fixes

        - Fixed support for MF H264 with Windows 7

- **Version 4.0.1 - 1 October 2019**
    - Improvements
        - iOS support!
            - New platform plugin, rock solid, written from scratch
            - Highly optimised and lightweight for professional use
            - Hardware accelerated video encoding
            - Supports H.264 and HEVC in MP4, M4V and MOV containers
            - Supports the Metal graphics API
            - Unity 2017 - 2019 supported
    - Bug fixes
        - Fixed a compiler warning in Unity 2019.2 related use of deprecated AudioSpeakerMode.Raw enum

- **Version 4.0.0 - 24 July 2019**
    - Improvements
        - macOS support!
            - New platform plugin, rock solid, written from scratch
            - Highly optimised and lightweight for professional use
            - GPU hardware accelerated video encoding (when supported)
            - Many video formats including H.264 and HEVC, in both MP4 and MOV containers
            - Supporting both the OpenGL and Metal graphics APIs
            - IL2CPP supported
            - Unity 5.6 - 2019 supported
        - Windows MF H.264 codec performance improvements
        - Improved performance for 360 mono captures
        - Added Desktop as output folder option
        - Added assembly definitions (asmdef files) for modularity and improved script compile times
        - Code improvements via refactoring
    - Changes
        - Removed support for Unity 4.x
        - Removed support for Unity 5.5.x and below

- **Version 3.7.4 - 14 June 2019**
    - Bug fixes
        - Fixed rare crash bug reported on Intel HD Graphics 620
        - Added back previously removed call to Stop()

- **Version 3.7.2 - 17 May 2019**
    - Bug fixes
        - Fixed major crash bug that would sometimes happens when stopping a capture
        - Fixed a crash bug that could happen by quitting the application while a

capture is running

- **Version 3.7.0 - 21 February 2019**
  - Bug fixes
    - Fixed issue where certain effects (eg skinned meshes) would be captured incorrectly. We now wait for the end of the frame to capture. This can be disabled using the WaitForEndOfFrame property

- **Version 3.6.10 - 13 February 2019**
  - Improvements
    - Added support for codecs and microphone device names with unicode characters

- **Version 3.6.8 - 1 November 2018**
  - Bug fixes
    - Fixed a crash bug that sometimes happened when stopping the capture due to a threading issue

- **Version 3.6.7 - 4 September 2018**
  - Bug fixes
    - Fixed a memory leak that would happen at the end of each capture

- **Version 3.6.6 - 23 July 2018**
  - Improvements
    - Added faster and more accurate stereo 360 rendering via adding support for Unity 2018.1's stereo cubemap rendering to the CaptureFromCamera360 component

- **Version 3.6.5 - 9 July 2018**
  - Bug fixes
    - Fixed bug with audio capture of Unity audio (regression since 3.6.0)

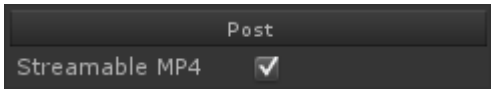
- **Version 3.6.4 - 7 July 2018**
  - Improvements
    - PNG exporter now supports exporting without alpha
  - Bug fixes
    - Fixed the PNG writer for high resolutions

- **Version 3.6.2 - 6 July 2018**
  - Improvements
    - Exported PNG files now more compatible (eg Photoshop After Effects, Premiere, Nuke etc)

- **Version 3.6.0 - 28 June 2018**
  - Improvements
    - Simultaneous captures are now supported (BETA)

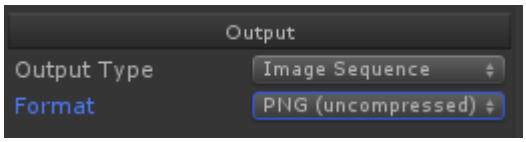
- Bug fixes
  - Fixed some codecs (such as MagicYUV) not being enumerated
  - Fixed errors when stopping a Stereo ODS capture from the Capture Window

- **Version 3.5.2 - 1 June 2018**
  - Improvements
    - MP4 encoding can now applies the “fast start” patch which moves the ‘moov’ atom to the start of the file. This allows these videos to be directly streamed from websites or servers



    - The Movie Capture window is now persistent allowing it to capture across scene changes
    - Added new persistent option to prevent CaptureFrom components being destroyed during scene loads
  - Bug fixes
    - Fixed instantiation issue where OnDestroy() would unload the entire plugin

- **Version 3.5.0 - 25 May 2018**
  - Improvements
    - Added image sequence output mode with fast PNG support



    - Moved all capture functions from Update() to LateUpdate() to fix issues where some game logic hadn’t been applied before the capture
  - Bug fixes
    - Fixed issue where using Final IK with the “Fix Transforms” field ticked would result in the IK not being applied
    - Fixed issue where first frame of wouldn’t capture when using timelapse
  - Other
    - Enabled compiler optimisations in native plugin
    - Many UI improvements
    - Corrected documentation steps for exporting transparent video

- **Version 3.4.4 - 23 May 2018**
  - Improvements
    - Added new timelapse field to allow videos to be captured at a fraction of the rate at which they will be played back. The field timelapseScale controls this. If your video frame rate is set to 30 and timelapseScale is set to 2 then it will capture only at 15fps, but generate a 30fps video
  - Bug fixes

        - Fixed rare issue where render functions could become null
        - Blocked FFDShow codec from enumerating as this old codec is quite buggy and could cause crashes
    - Other
        - Improved version number function

- **Version 3.4.2 - 17 May 2018**
    - Improvements
        - Added new 180 degree capture mode to the 360 and 360ODS capture components
    - Bug fixes
        - CaptureFromCamera component now includes contributing cameras that are disabled in case they are enabled later
        - Script execution order of capture components changed from 100 to 29000 so capture happens later than user scripts
    - Other
        - “View last capture” button now coloured to improve workflow

- **Version 3.4.0 - 14 May 2018**
    - Improvements
        - CaptureFromTexture now has logic to signal when the texture has changed, making texture capture more controllable, and capturing from webcam textures produce smoother results
        - CaptureFromCamera UI improved to make working with contributing cameras easier
        - API: videoCodecPriority field changed to a property so that it correctly calls SelectCodec() when changed
        - API: forceGpuFlush added to force the GPU to flush during each capture, which causes less latency but can reduce performance
        - Better codec enumeration so non-compressors are no longer included, and Logitech’s old video encoder is now excluded as it causes problems on some systems and isn’t generally useful
    - Bug fixes
        - Fixed issue when using MF H.264 codec where pausing didn’t properly pause the internal clock
    - Other
        - Webcam demo changed to only capture a single camera at once since multi-source capture isn’t working currently, and to use the new manual signal with CaptureFromTexture to make sure no frames are missing or duplicated

- **Version 3.3.2 - 23 February 2018**
    - Improvements
        - UnityAudioCapture component can now be attached to other objects besides the AudioListener, such as the AudioSource component, and it can mute the audio once it has grabbed it

    - Bug fixes
        - Fixed a bug where RenderTextures with generate mipmaps enabled couldn't be captured
        - Fixed 360 stereo ODS capture problem with some image effects (such as HBAO) which needed reactivation
        - Fixed some shaders that had been automatically upgraded for newer versions of Unity so they're backwardly compatible again
    - Other
        - Updated all copyright messages to 2018
        - Tested with Unity 2018.1.0b8
        - Now also uploading versions using Unity 5.5 and 5.6 to prevent the script/shader upgrade warnings

- **Version 3.3.1 - 10 December 2017**
    - Bug fixes
        - Fixed shader compile error in Unity 2017.3 when single pass instanced stereo mode is enabled for VR rendering

- **Version 3.3 - 9 November 2017**
    - Improvements
        - Major performance improvements by no longer stalling the GPU during capturing
        - The mouse cursor now has an offset value for the hotspot and a scale slider

- **Version 3.2 - 5 October 2017**
    - Improvements
        - CaptureFromCamera now supports camera chains where multiple cameras contribute to the final render
        - Improved compatibility with older systems by not relying on some Media Foundation DLLs
        - Added a new option "_supportTextureRecreate" which now defaults to "off" which improves performance by removing a per-frame GPU sync point. Enable this option if you need to support texture recreation during a capture (for example when the GPU device is lost due to window resize or alt-tab when using exclusive fullscreen mode)
        - Improved inspector UI for audio source selection
        - Added MF H.264 capture option to CaptureGUI
    - Bug fixes
        - Fixed some ODS 360 capture settings not being applied

- **Version 3.14 - 7 September 2017**
    - Improvements
        - Added new UI button to view the captured video immediately instead of via Windows Explorer
        - Improved support for capturing when using OpenGL emulation, which

can happen when having the build platform not set to Windows Standalone
        - CaptureFromScreen component generates less garbage by caching the YieldInstruction
        - Improved encoded time display
        - Improved error messages
        - Improvements to UI
    - Bug fixes
        - Fixed bug where capture window could not set the file name or extension
        - Fixed bug where Capture Window would add UnityAudioOutput component to the main camera instead of to the GameObject with the AudioListerner
        - Fixed the encoded time information for real-time captures

- **Version 3.12 - 30 August 2017**
    - Bug fixes
        - CaptureFromTexture and CaptureFromCamera360 components now no longer accumulate and render when capture is paused
    - Minor
        - Added helpful warning when using the profiler with 360 Camera ODS component. Have the profiler completely disabled (disable record, close the window, remove the tab and restart Unity), otherwise it will use too much memory when using ODS component.
        - Added new “about” section to inspector
        - Added a better check for plugin version mismatches
        - Some improved texture cleanup and discards
        - Improved camera 360 demo script to make cubes fall away instead of disappear

- **Version 3.10 - 27 August 2017**
    - Upgrade Notes
        - If upgrading from version 2.x then you must delete all of the files for any previous copies of AVPro Movie Capture in your project before importing this version. This is because many files have been renamed and the folder structure has changed significantly.
    - Improvements
        - A preview of the captured frames is now displayed
        - Pause during offline rendering now pauses Unity by settings Time.timeScale to zero
        - Various minor UI and other improvements
    - Bug fixes
        - Fixed deferred rendering support for stereo ODS component
        - Fixed motion blur timing and sample count issues
        - Fixed disabling MSAA when using deferred rendering with stereo ODS component

        - Fixed some anti-aliasing and cubemap depth states not being set sometimes
        - Added a workaround for a bug in Unity 5.6.0 - 5.6.1 (#902409)
        - Noted a bug in Unity 2017.1.0 - 2017.2.0 in CaptureFromCamera360 that causes the background to render white when camera clear mode is set to solid colour

- **Version 3.01 - 21 August 2017**
    - Upgrade Notes
        - If upgrading from version 2.x then you must delete all of the files for any previous copies of AVPro Movie Capture in your project before importing this version. This is because many files have been renamed and the folder structure has changed significantly.
    - Improvements
        - Added named pipe output option to the Capture Window and improvements to the inspector
    - Bug fixes
        - Fixed some bugs related to using named pipes
        - Fixed some rare null exceptions that could happen when upgrading from previous version which used specific codec indices

- **Version 3.0 - 11 August 2017**
    - Upgrade Notes
        - If upgrading from version 2.x then you must delete all of the files for any previous copies of AVPro Movie Capture in your project before importing this version. This is because many files have been renamed and the folder structure has changed significantly.
    - Improvements
        - Added Unity 2017 support
        - UI revamped to be easier to use and better looking
        - Huge code cleanup: moved everything into a namespace, classes renamed, code documentation improved, documentation updated etc
        - Added new Omni-Directional Stereo (ODS) capture component for capturing stereo 360 in a more accurate way. This new method is much slower and is intended for off-line rendering only. Based on: https://developers.google.com/vr/jump/rendering-ods-content.pdf
        - Added options to automatically stop the capture / render after a certain number of frames or seconds
        - Performance improvements, the capture no longer forces Unity to run at the capture rate
        - Exposed the Camera360 stereo options in the AVPro Movie Capture window
        - CaptureFromCamera and CaptureFromCamera360 components no longer require to be attached to a GameObject with a Camera. The camera can be set manually via the exposed field. If no camera is assigned it will try to find the Camera component on the same

GameObject.
        - The free trial version is now on the Unity Asset Store and allows for 10 seconds of capture without any watermarking.
    - Bug fixes
        - Fixed audio capture in Camera360 component
        - Fixed captures trying to use MSAA and Deferred rendering together
        - Fixed a bug where “Start Paused” option would be ignored
        - Fixed issue where UnityAudioCapture component would be enabled when capture started with StartPaused option and Pause/Resume wouldn’t toggle enabled status of UnityAudioCapture component
        - Fixed “Open Last Capture” sometimes not storing the last capture

- **Version 2.99 - 11 April 2017**
    - Improvements
        - Added stereo rendering support to the 360 camera capture component. 360 captures can now be rendered in mono, stereo top-bottom or stereo left-right
    - Bug fixes
        - Fixed a memory leak in the Media Foundation recorder

- **Version 2.98 - 7 April 2017**
    - Improvements
        - Texture capture component is now much faster
        - Motion blur capture now works with the Surround 360 and Texture capture components
        - Added option to include mouse cursor in the capture
        - Texture capture demo improved
        - Increased capture resolution limit from 8K to 16K
    - Bug fixes
        - Fixed audio delay when capturing webcam and microphone
        - Removed dependency on Media Foundation so older versions of Windows are supported again
        - Various fixes made to the motion blur component

- **Version 2.96 - 6 March 2017**
    - Improvements
        - 360 surround capture can now render uGUI elements when canvas is set to camera or world modes.
        - 360 surround capture now supports camera rotation
        - Fixed 360 surround capture orientation, as it was flipped horizontally
        - Improved motion blur support in 360 camera capture
        - Added Media Foundation capture option to Editor window
    - UI / Workflow
        - Editor window UI improvements, including context hiding and usage warnings
    - Bug fixes

        - Fixed FromCamera capture bug where build captures would render black to the screen
        - Fixed bug where editor window would unnecessarily spawn AudioCapture component
        - Fixed bug where audio capture component would be accessed when it shouldn't
        - Fixed crash bug in webcam demo

- **Version 2.94 - 27 September 2016**
    - Fixed issue where camera captures would freeze when vsync changed

- **Version 2.92 - 26 September 2016**
    - Fixed support for capturing transparency
    - Improved robustness for enumeration of buggy audio devices
    - Update documentation

- **Version 2.90 - 16 September 2016**
    - New Features
        - New 360 degree equi-rectangular capture for VR
        - Better support for VR capture
        - Real motion blur feature for added quality to offline captures (BETA)
        - Captures from Camera can now be higher resolution than displayed allowing capture resolutions over 3840x2160
        - Captures from Camera can now have custom anti-aliasing settings
        - Experimental Media Foundation H.264 (MP4) encoder
    - Changes
        - Added 75, 90 and 120 FPS capture modes for VR
        - Added variables to disable vsync and frame rate changing
        - Stop function now has parameter to skip processing of any pending frames - useful for canceling a capture quickly. Also added new CancelCapture() function which stops capturing fast and deletes the capture file.
        - Better capture of physics simulations in off-line capture mode
        - Now stops recording when disk space is too low instead of crashing
        - Added setting to start capturing in a paused state
        - Dropped support for older versions of Unity, project now built with Unity 4.6
        - Tested up to Unity 5.4
        - Project files rearranged
    - UI / Workflow
        - Editor window layout improvements
        - Component editor layout improvements
    - Optimisations
        - Capture from Camera and RenderTexture much faster
    - Bug fixes
        - Fixed linear color-space capture issues

        - Fixed crash bug on 64-bit builds when creating lots of captures
        - Fixed audio recording latency issue when using pause/resume
        - Fixed audio capture from Unity non-threaded issue causing clicks in recordings
        - Fixed codecs not showing config options sometimes

- **Version 2.72 - Monday 15 June 2015**
    - Fixed Unity 5.1 support

- **Version 2.70 - Wednesday 27 May 2015**
    - Fixed 64-bit VFW video codec enumeration issue
    - Fixed issue where multiple codecs could not be used at once
    - Improved demo capture dialog so it shows recording stats
    - Improved webcam demo to allow recording of multiple webcams
    - Fixed crash bug in watermarking

- **Version 2.64 - Saturday 18th April 2015**
    - Unity 4.6 and 5.0 support added
    - DX11 GPU red-blue swap and downscaling
    - Dropped official Unity 3.x support, minimum version supported is now 4.0
    - Fixed audio recording freeze bug
    - Better plugin folder structure
    - Faster watermarking
    - Added experimental writing to ‘named pipes’

- **Version 2.55 - Monday 28 July 2014**
    - Unity 4.5 support
    - Unity audio capture now supports speaker modes other than stereo
    - Improved error reporting
    - Improved vsync handling

- **Version 2.52 - Thursday 27 February 2014**
    - Fixed unnamed scene recording bug
    - Removed usage of deprecated Unity functions
    - Added display of output frame resolution to UI
    - Video file now deleted before recording to prevent AVI bloat

- **Version 2.5 - Monday 24 February 2014**
    - Added new trial version
    - Added script to automatically copy the DLL files during installation
    - Added better output path options
    - Added support for more YCbCr formats
    - Improved in-editor GUI to allow capturing from specific camera
    - Various minor fixes to the demos
    - Fixed some script issues on non Windows platforms

- **Version 2.48 - Wednesday 15 January 2014**
    - Fixed crash bug when system has no codecs or devices
    - Minor improvements to UI

- **Version 2.46 - Monday 8 January 2014**
    - Added recording stats (file size and time)
    - Improved recording stats GUI
    - Fixed editor launch crash bug when using non Windows built settings
    - Added OpenGL emulation notes to FAQ for captures appearing upside-down

- **Version 2.44 - Monday 6 January 2014**
    - Fixed editor error message when launching configure dialogs

- **Version 2.42 - Monday 30 December 2013**
    - Fixed y-flip bug
    - Fixed camera capture bug using wrong texture size
    - Fixed DX11 viewport capture bug

- **Version 2.4 - Friday 27 December 2013**
    - Added codec configuration dialog support detection
    - Reduced CPU usage of new EditorWindow widget
    - Improved codec lists in new EditorWindow widget
    - EditorWindow can now play the scene to start recording
    - Improved documentation

- **Version 2.3 - Wednesday 4 December 2013**
    - Added proper fast native DX11 support
    - Scene capture no longer requires a camera
    - Added new EditorWindow widget for easier capturing from Editor
    - Improved documentation for best codec settings

- **Version 2.2 - Thursday 22 August 2013**
    - Added support for writing to MP4 container
    - Added support for Unity 4.2
    - Added automatic DLL swap for 64-bit builds
    - Fixed some material leaks
    - Fixed DX11 colour swap and flip bug

- **Version 2.02 - Monday 18 March 2013**
    - Added Unity 4.1 support
    - Fixed some platform #if issues

- **Version 2.0 - Monday 12 March 2013**
    - Added audio recording directly from Unity
    - Fixed GL.IssuePluginEvent() conflict bug with other AVPro plugins

    - Fixed DX11 recording in Unity 4.0
    - Fixed bug in audio codec listing
    - Renamed and restructured code

- **Version 1.8 - Tuesday 18 December 2012**
    - Added audio codec enumeration
    - Added Unity 4.0 support
    - GUI improved
    - Added more demos

- **Version 1.6 - Thursday 6 September 2012**
    - Added ability to pause movie capture
    - Scene capture resolution can differ from Screen resolution
    - Inspector: displays capture rate and has buttons to control capture
    - Less CPU usage
    - Optimisation: removed software RB channel swap
    - Optimisation: removed per-frame memcpy
    - Optimisation: removed vertical flip
    - GUI layout improved
    - Lots of source cleaning up

- **Version 1.5 - Monday 6 June 2012**
    - Improved smoothness of captures significantly.
    - 64-bit Windows support added.
    - Added GUI to easily set up recordings (taken from previous demo scene).
    - Added code to detect dropped frames during encoding.

- **Version 1.4 - Thursday 15 March 2012**
    - Much faster capturing due to new Unity 3.5 native API features.

- **Version 1.3 - Friday 17 February 2012**
    - Added audio for testing audio recording.
    - Autodetection of loopback audio devices.

- **Version 1.2 - Saturday 4 February 2012**
    - Added ability to capture GUI.
    - Added audio capture.
    - Added resizing to half, quarter, eighth resolution.
    - Improved capture performance and smoothness by only preparing the frame capture data when the encoder requires it.
    - Automatic disable of vsync helps performance.
    - Rounding to multiple of 4 resolution to help codec compatibility.
    - Added ability to set target frame rate (15, 24, 30, 60).
    - Added ability to set own file name.
    - Video codecs can now be configured.
    - Fixed various minor bugs.

- **Version 1.1 - Tuesday 24 January 2012**
    - Removed Vista/Win7 dependency (WMV).

- **Version 1.0 - Thursday 17 January 2012**
    - Initial release submitted to Asset Store.

Appendix C - Road Map

- **Version 5.x -?**
    - Capture only a region of the screen/texture
    - Export depth and other deferred channels

- **Version 5.x - ?**
    - ← Your suggestion here
    - Improve code documentation?
    - Recompile with newer version of VS?
    - Super-resolution for “screen” capture?
    - Fix H.264 initial freeze bug?
    - Support floating point audio to integer conversion?
    - Add audio delay support?
    - Bicubic downscaling?
    - Investigate windowed mode and vsync-on capture bugs
    - Embed DirectShow MP4/MOV container support
    - GPU context manager
    - GPU context per recorder
    - Commandbuffer support for ordering to exclude IMGUI
    - Motion blur demo scene