



# Flashback Video Recorder

## Version 1.2

The latest version of this document can be found at  
<http://www.launchpointgames.com/unity/flashback>

**Created by LaunchPoint Games**

Contact: [support@launchpointgames.com](mailto:support@launchpointgames.com)

## Table of Contents

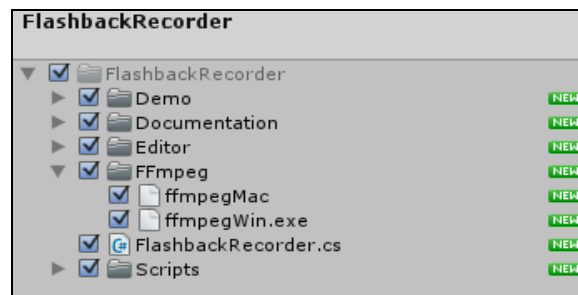
1. Introduction and Overview .....	3
2. Key Limitations.....	4
3. Demo Quick Start .....	5
4. Flashback Recorder Integration Guide.....	6
Step 1: Import the Flashback Video Recorder Package.....	6
Step 2: Attach the Flashback Video Recorder to the Camera.....	6
Step 3: Configure the Flashback Recorder's Properties.....	7
Step 4: Using the Flashback Recorder.....	9
Step 5: Preparing a Standalone Build.....	10
5. Code API.....	11
FlashbackRecorder API .....	11
API Examples .....	15
6. FAQ and Troubleshooting .....	17
How does Flashback Video Recorder work?.....	17
What is FFmpeg and why is it included with Flashback Video Recorder?.....	17
How can I improve performance when using Flashback Video Recorder? .....	17
Why do I get a QuickTime error when importing MP4 files? .....	18
Why does my captured video look jittery? .....	18
How can I reduce the file size of my captured GIFs? .....	19
Why do my GIFs appear low quality?.....	19
Can I capture more than 30 frames per second?.....	19
Why is video capture not working? .....	20
Why do shadows disappear when I'm using Flashback Video Recorder? .....	21
What if I have improvements or new features for Flashback Video Recorder? .....	21
7. Features Roadmap.....	22
8. Terms & Conditions .....	23

## 1. Introduction and Overview

Flashback Video Recorder continuously captures the last few moments of gameplay, which can be written to a GIF or an MPEG-4 file with the press of a button. Only desktop platforms (Windows PC and Mac OSX) are currently supported, and all encoding is done through FFmpeg.

FFmpeg is a third party, open source, cross-platform tool that lets you easily convert video formats, and is bundled with Flashback Video Recorder. You can learn more here: <http://ffmpeg.org>

When you import Flashback Video Recorder into your project, the following assets will be included:



Demo/	Stores the scene file and all other assets for a fully functional demonstration of Flashback Video Recorder. When importing to an existing project, you may choose to exclude this folder.
Documentation/	Holds the latest PDF version of this file.
Editor/	Contains helper scripts and resources used in the Unity Editor and Inspector window.
FFmpeg/	Contains the FFmpeg binaries for Windows and Mac OSX. If you are only building for one target platform, you can exclude the file you don't need.
FlashbackRecorder.cs	The interface to Flashback Video Recorder's core logic. You will attach this to your main camera.
Scripts/	Contains helper scripts that launch FFmpeg with the settings specified.

This guide covers adding Flashback Video Recorder to a new or existing project, and provides a detailed explanation on how the package works under the hood.

If you are not sure if Flashback Video Recorder is right for your project, check out the “Key Limitations” section of this document, as well as the “Features Roadmap” section. If you have additional questions, please email us directly at [support@launchpointgames.com](mailto:support@launchpointgames.com). We do our best to respond as quickly as possible.

## 2. Key Limitations

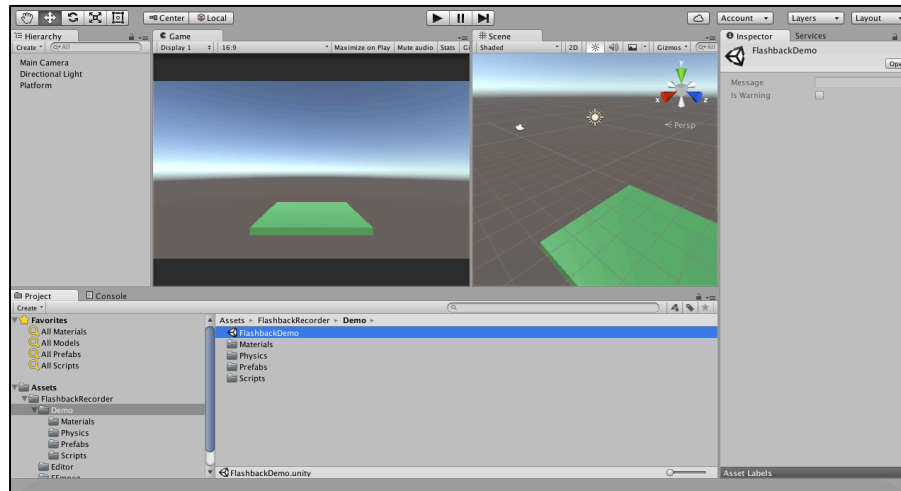
There are some limitations that you should be aware of when deciding whether Flashback Video Recorder is appropriate for your project.

Limitation	Description
Desktop Only	Flashback Video Recorder only works on Windows and Mac standalone, or in the editor. Currently, there is no plan to support mobile devices.
Performance Impact	Flashback Video Recorder will have a minor impact on performance; up to 2-5 ms per frame depending on capture settings and computer hardware. See the “FAQ and Troubleshooting” section for ways to improve performance.

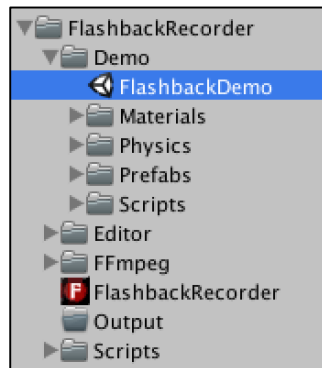
To learn more about planned features, check out the “Features Roadmap” section of this document.

### 3. Demo Quick Start

Flashback Video Recorder comes with a demo scene you can use for testing. Start by creating a new project and importing all files in the Flashback Video Recorder asset package.

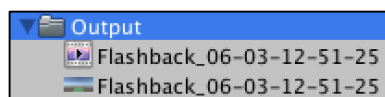


Step 1. Open the Demo Scene located in /Assets/FlashbackRecorder/Demo/



Step 2. Play in editor, and after a few seconds, press the SPACE BAR.

Step 3. The recorded GIF and MP4 files are saved with the prefix “Flashback” in the following project folder: /Assets/FlashbackRecorder/Output/



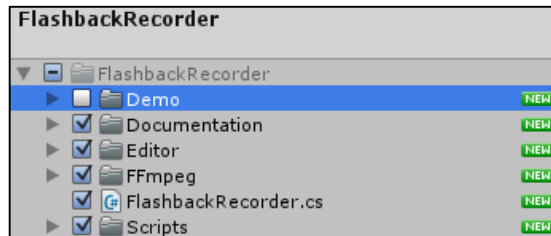
**Note:** You may have to right-click in the file browser and select “Refresh” for the files to appear in the Unity Project view.

On Windows, you may get an error when importing MP4 files into Unity if QuickTime is not installed. This is expected, and you can ignore the error. There is more information in the “FAQ and Troubleshooting” section of this document.

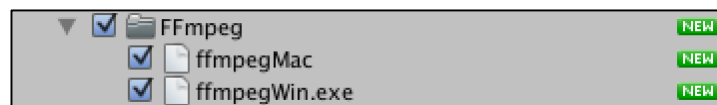
## 4. Flashback Recorder Integration Guide

### Step 1: Import the Flashback Video Recorder Package

Import the Flashback Video Recorder package into your project. If you are working with an existing project, you can choose to exclude the "Demo" folder:



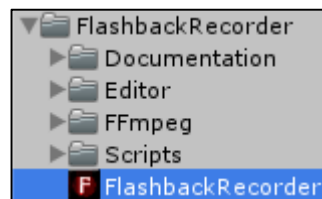
If you are planning to build for only one platform (either Windows or Mac), then you can exclude the FFmpeg file that you won't be using. If you are unsure, you can keep both files:



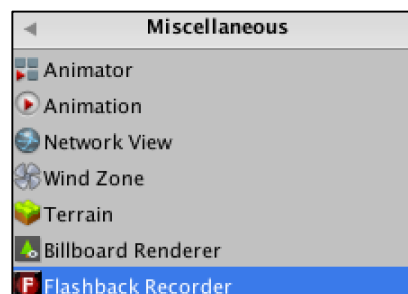
### Step 2: Attach the Flashback Video Recorder to the Camera

After you have imported the Flashback Recorder package, you will need to attach the FlashbackRecorder script to the main camera. There are two ways to do this:

Option 1. Drag the "FlashbackRecorder/FlashbackRecorder.cs" script onto the camera object.

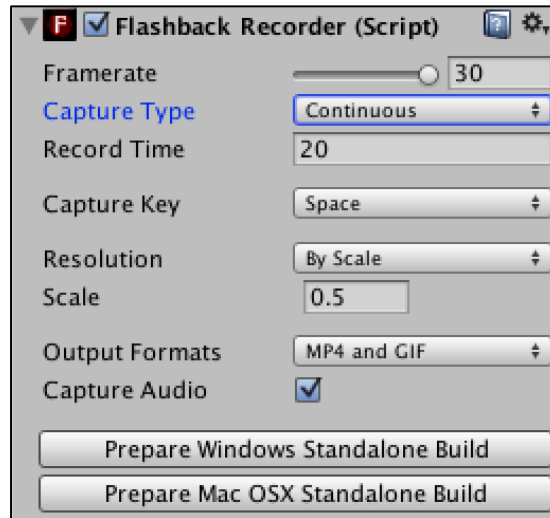


Option 2. Click on the camera object in the Hierarchy window, select "Add Component" then click "Miscellaneous" and "Flashback Recorder".



### Step 3: Configure the Flashback Recorder's Properties

After you've attached the FlashbackRecorder component to a camera, the inspector allows you to set the following properties:



**Framerate** – The number of frames to capture every second. This should be lower than your actual camera framerate, otherwise you'll see stuttering in the output video. Lower values will result in smaller file sizes, but decreased quality. By default, you can capture up to 30 frames per second.

**Capture Type** – You can choose to capture continuously or toggle recording on/off. If set to “Continuous”, the last few seconds (as specified by “Record Time”) will be written to disk when the “Capture Key” is pressed. If the Capture Type is set to “Toggle”, the first time the “Capture Key” is pressed, recording will start, the second time it is pressed, the recording will stop and the output will be written to disk.

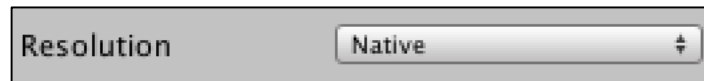
**Record Time** – The amount of video history to save to disk (in seconds) if the Capture Type is set to “Continuous”. Be aware that larger values will require more memory and will take longer to process when writing to disk.

**Capture Key** - The key you want to assign to video capture. By default, it is the SPACE BAR, but you can select any key on the keyboard. If you'd like to implement your own method of capturing the video output, set the Capture Key to “None” and then call "SaveCapturedFrames" in the FlashbackRecorder component directly.

**Resolution** – There are three different ways to specify the output resolution of the captured video, each with different options.

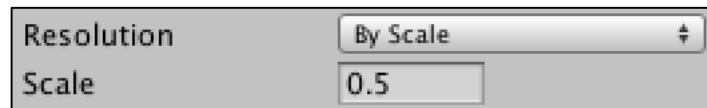
Keep in mind that no matter which option you choose, the higher the resolution, the more memory and processing time is required.

*Native* – The video will be captured at the screen’s current (native) resolution. The height and width of the video will be set at runtime during the `Start()` function call.



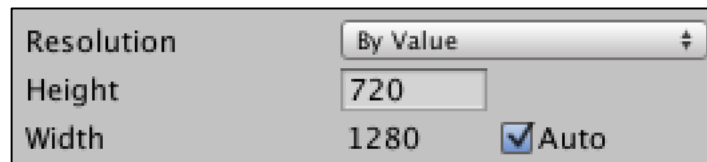
A control panel with a label 'Resolution' on the left and a dropdown menu on the right. The dropdown menu is open, showing 'Native' as the selected option.

*By Scale* – The video will be scaled from the native resolution. A value of 1 will match the screen’s height and width (which is the same as choosing *Native*), a value of 0.5 will be half the screen’s height and width, and a value of 2 will be twice the native height and width.



A control panel with a label 'Resolution' on the left and a dropdown menu on the right. The dropdown menu is open, showing 'By Scale' as the selected option. Below the dropdown is a text input field containing the value '0.5'.

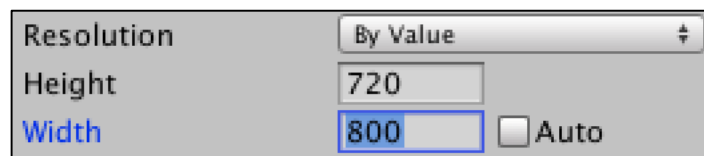
*By Value* – This gives you the option to explicitly specify the pixel height of the captured video, and optionally, the width. By default, the width is automatically calculated to maintain the camera’s aspect ratio.



A control panel with a label 'Resolution' on the left and a dropdown menu on the right. The dropdown menu is open, showing 'By Value' as the selected option. Below the dropdown are two text input fields: 'Height' with the value '720' and 'Width' with the value '1280'. To the right of the 'Width' field is a checkbox labeled 'Auto' which is checked.

If “Auto” is selected, the width will be calculated at be set at runtime during the `Start()` function call.

You can always manually choose to set the width by unchecking “Auto”.



A control panel with a label 'Resolution' on the left and a dropdown menu on the right. The dropdown menu is open, showing 'By Value' as the selected option. Below the dropdown are two text input fields: 'Height' with the value '720' and 'Width' with the value '800'. To the right of the 'Width' field is a checkbox labeled 'Auto' which is unchecked.

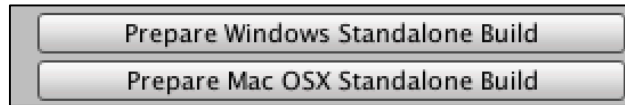
**Output Formats** – You can choose the type of file that will be saved to disk when the user presses the “Capture Key”: either a GIF file, an MP4 file, or both. The capture framerate and resolution will have a direct impact on the time it takes to process the output files, and the disk space required.

In general, GIF files will be much larger than MP4 files, and will have a limited color palette. If you are only interested in capturing GIF files, set a lower framerate (15 or lower), and a low resolution (By Scale set to 0.25, or By Value with a height less than 400 pixels or so).



**Capture Audio** – This option will only appear if the output formats include MP4 video. If selected, the video file will include the in-game audio track.

**Prepare a Windows or Mac OSX Standalone Build** – Before building for a specified target platform, make sure you click the appropriate button in the Flashback Recorder inspector window:

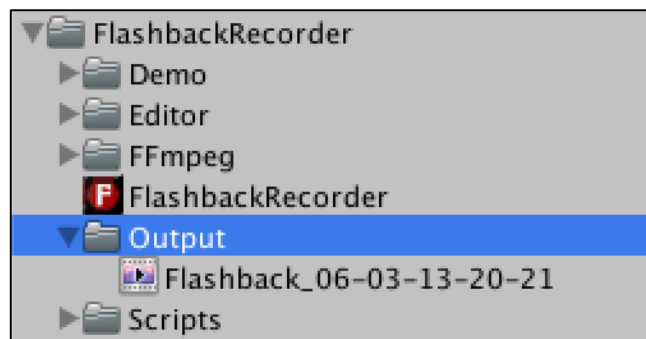


This will deploy the FFmpeg file to the StreamingAssets folder, so that it is included with your build, and so video can be captured at runtime. See Step 5 for more detail.

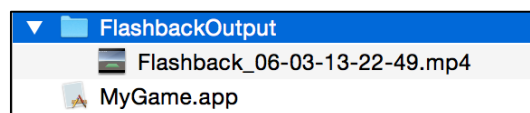
#### Step 4: Using the Flashback Recorder

When running the game, either in the editor or in a standalone build, the specified amount of video history will be stored in memory. The captured frames will be written to disk when the user presses the button associated with the “Capture Key”.

By default, when running in the editor all files will be saved in the "Assets/FlashbackRecorder/Output/" folder in your project root folder with the prefix "Flashback\_".



If running in a standalone build, files are saved in a folder called "FlashbackOutput" at the root level of your executable (or .app file on a Mac).

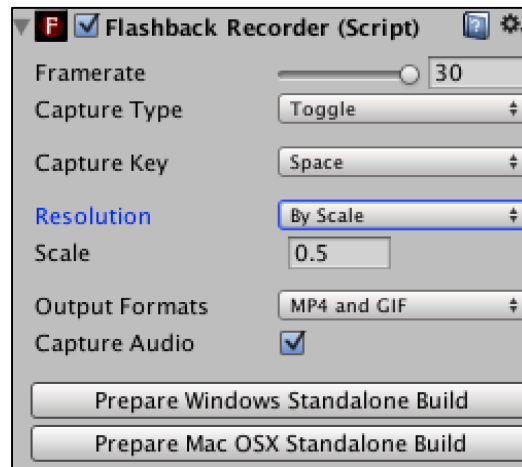


You can change this programmatically using the API. See the “Code API” section for more details.

## Step 5: Preparing a Standalone Build

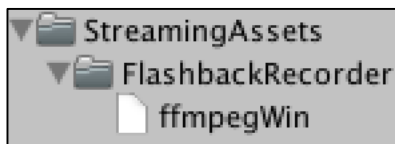
To run a standalone build with Flashback Video Recorder, you will need to make sure the FFmpeg binary is accessible at runtime. To do this, the appropriate file (either FFmpegWin.exe for Windows or FFmpegMac for OSX) must be in the StreamingAssets/FlashbackRecorder folder when you build your game.

There is a simple way to deploy the correct asset – before you build your game, click on “Prepare Windows Standalone Build” or “Prepare Mac OSX Standalone Build” as appropriate.

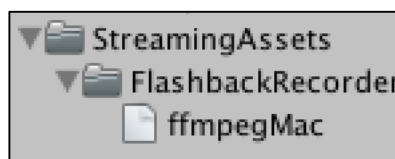


Clicking either of these buttons will first delete the /Assets/StreamingAssets/FlashbackRecorder folder (if it exists) so that the wrong binary won't be included with the build, which would needlessly increase your game folder size. Then the appropriate FFmpeg file will be copied from Assets/FlashbackRecorder/FFmpeg to the StreamingAssets folder.

After preparing a Windows standalone build, you will see the following folder structure on your PC:



Or you will see the following folder structure on your Mac:



## 5. Code API

The core functionality of the Flashback Video Recorder is exposed through the FlashbackRecorder class. This allows you to make changes to the configuration at runtime, and capture video files based on events in your game.

### FlashbackRecorder API

The following is a list of all publically accessible methods in the FlashbackRecorder class.

Event Delegate	Description
<code>OnFlashbackFileCreated(string file)</code>	Executed after a file has been written to disk. Ex: <code>recorder.OnFlashbackFileCreated += MyDelegate;</code>

Enumerators	Description
<code>ResolutionType {Native, Scale, Manual}</code>	Describes how the capture resolution was set. <i>Native</i> for using the Screen height and width. <i>Scale</i> for resizing the native resolution. <i>Manual</i> for explicitly setting the height and width using pixel values.

Accessors	Description
<code>int GetFrameRate()</code>	The number of frames captured every second
<code>int GetNumberOfCapturedFrames()</code>	The total number of captured frames
<code>float GetMaxCaptureTime()</code>	The max allowed capture history, in seconds
<code>float GetCurrentCaptureTime()</code>	The current amount of capture history, in seconds
<code>KeyCode GetCaptureKey()</code>	The key code that, when pressed, will save the captured frames to disk
<code>bool CaptureAudio()</code>	True if the audio is being captured
<code>bool ContinuousCapture()</code>	True if frames are continuously being captured (ie, Toggle is not selected)
<code>ResolutionType GetResolutionType()</code>	Returns <i>Native</i> , <i>Scale</i> , or <i>Manual</i> depending on how the capture resolution was determined
<code>int GetHeight()</code>	The capture resolution height in pixels

<code>int GetWidth()</code>	The capture resolution width in pixels
<code>float GetResolutionScale()</code>	How much the resolution was scaled (ex: 0.5 if the output is half the size of the native resolution). Dynamically set unless the <code>ResolutionType</code> is set to <i>Scale</i>
<code>bool AutomaticWidth()</code>	True if the Width was calculated automatically. Can only be true if the <code>ResolutionType</code> is set to <i>Manual</i>
<code>int GetNumberOfPendingFiles()</code>	The number of files that are being written to disk
<code>bool SaveMP4()</code>	True if the output will be written to an MP4 file
<code>bool SaveGIF()</code>	True if the output will be written to a GIF file

Recorder Interface	Description
<code>void SetCaptureKey(KeyCode key)</code>	Set the keyboard key associated with file creation. Pass in <code>KeyCode.None</code> if you wish to call <code>SaveCapturedFrames</code> directly.
<code>void SetContinuousCapture(bool continuous)</code>	If true, frames will be automatically captured. If false, the recorder will need to be toggled on/off.
<code>void SetResolutionToNative()</code>	Sets the capture resolution to the native screen size. Requires that the frame buffer be cleared (i.e., all captured history will be lost).
<code>void SetResolutionByScale(float scale)</code>	Sets the capture resolution to the native screen size multiplied by the specified scale. Requires that the frame buffer be cleared (i.e., all captured history will be lost). Ex: <code>recorder.SetResolutionByScale(0.5f);</code>
<code>void SetResolutionByValue(int height, int width)</code>	Allows you to explicitly set the capture resolution. The frame will be rendered, then scaled. If the width is set to -1, then it will automatically be calculated to match the camera's aspect ratio. Ex: <code>recorder.SetResolutionByValue(720, -1);</code>
<code>string GetRootFolder()</code>	The root folder of the game binary at runtime. Can be used to explicitly set the file output directory.
<code>void SetOutputDirectory(string dir)</code>	Sets the base directory where all captured videos will be stored. Ex: <code>recorder.SetOutputDirectory(recorder.GetRootFolder() + "myCaptureFolder/");</code>
<code>void ClearStoredFrames()</code>	Clears the frame buffer history.

<code>void UpdateCaptureSettings(float captureTime, int framerate)</code>	Sets the frames per second that are captured and how much video history is stored. Requires that the frame buffer be cleared (i.e., all captured history will be lost).
<code>void CaptureAudio(bool saveAudio)</code>	Sets whether the audio will be recorded and saved with the video output (only for MP4)
<code>void SaveMP4(bool saveMP4)</code>	Sets whether an MP4 file will be written to disk.
<code>void SaveGIF(bool saveGIF)</code>	Sets whether a GIF file will be written to disk.
<code>bool StartCapture()</code>	If <code>ContinuousCapture()</code> returns false, toggles video capture on. Returns true if capture started successfully.
<code>bool StopCapture()</code>	If <code>ContinuousCapture()</code> returns false, toggles video capture off. Returns true if capture was stopped.
<code>bool IsCapturingVideo()</code>	Returns true if frames are currently being stored. ie, if <code>ContinuousCapture()</code> is true or if video capture was toggled on through <code>StartCapture()</code>
<code>bool CanSaveCapturedFrames()</code>	When a file is being written to disk, a new video file cannot be saved. Returns true if <code>SaveCapturedFrames(...)</code> will execute.
<code>bool SaveCapturedFrames()</code>	Will write the captured frames to files using an automatically generated name following the format: "Flashback_MM-dd-HH-mm-ss". Ex: "Flashback_06-03-14-45-39.mp4" Returns true if the file creation process was started.
<code>bool SaveCapturedFrames(string filename)</code>	Will write the captured frames to files using the specified file name. Do not include an extension, since it will be set automatically based on the output formats. Ex: <code>recorder.SaveCapturedFrames("myFile");</code> Returns true if the file creation process was started.

<b>FFmpeg File Management</b>	<b>Description</b>
<code>void ConfigureFFmpegDirectories()</code>	Called by the FlashbackRecorder and the FlashbackEditor to set up the locations for the FFmpeg binaries. You should never have to call this explicitly.
<code>string FFmpegPackageDir()</code>	Path to the FFmpeg binaries in the Flashback package.
<code>string FFmpegStandaloneDir()</code>	Path to the FFmpeg binaries when running standalone.
<code>string FFmpegWin</code>	The name of the Windows FFmpeg binary. Defaults to "ffmpegWin.exe"

<code>string FFmpegMac</code>	The name of the Mac FFmpeg binary. Defaults to "ffmpegMac"
-------------------------------	--

## API Examples

```
//Example 1
//Waits for a specified amount of time before saving the video so that
// the moment of interest is in the middle, not at the end, of the
// captured video
//You can set the FlashbackRecorder Capture Key to "None" in
// the Inspector so the video isn't captured twice.
```

```
void Update(){
    if( Input.GetKeyDown(KeyCode.Space) )
        StartCoroutine( SaveVideo(2.0) );
}

IEnumerator SaveVideo(float delay){
    yield return new WaitForSeconds (delay);
    rec = GetComponent<FlashbackRecorder> ();
    rec.SaveCapturedFrames("myFileName");
}
```

```
//Example 2
//Change the video history to 14 seconds at 30 frames per second
FlashbackRecorder rec = GetComponent<FlashbackRecorder> ();
rec.UpdateCaptureSettings(14, 30);
```

```
//Example 3
//Implementing the OnFlashbackFileCreated delegate

void Start(){
    FlashbackRecorder rec = GetComponent<FlashbackRecorder> ();
    rec.OnFlashbackFileCreated += FileCreated;
}

void OnDestroy(){
    FlashbackRecorder rec = GetComponent<FlashbackRecorder> ();
    rec.OnFlashbackFileCreated -= FileCreated;
}

public void FileCreated(string file){
    Debug.Log("File Created: " + file);
}
```

```
//Example 4
//Change the capture resolution to 100 pixels high, and preserve
// the screen aspect ratio
FlashbackRecorder rec = GetComponent<FlashbackRecorder> ();
rec.SetResolutionByValue (100, -1);
```

```

//Example 5
//Set the capture resolution to 25% of the screen dimensions
FlashbackRecorder rec = GetComponent<FlashbackRecorder> ();
rec.SetResolutionByScale (0.25f);

//Example 6
//Change the root directory and capture a file with a specific name.
// In this case: "C:/myfolder/videos/myVideo.mp4"

void CaptureVideo(){
    string path = "C:/myfolder/videos/";
    string videoName = "myVideo";

    FlashbackRecorder rec = GetComponent<FlashbackRecorder> ();
    rec.SetOutputDirectory(path);
    rec.SaveGIF(false);
    rec.SaveMP4(true);
    rec.SaveCapturedFrames(videoName);
}

```



## 6. FAQ and Troubleshooting

### How does Flashback Video Recorder work?

There are three distinct steps to how Flashback Video Recorder captures and saves video output:

<i>Step</i>	<b>Capture</b>	<b>Preprocessing</b>	<b>File Output</b>
<i>Timeframe</i>	Continuously	On Capture Key Press	After Preprocessing
<i>Description</i>	Based on the capture framerate, the recorder saves the raw image data to memory. Each image is captured, scaled, and converted to a byte array to be saved in a queue.	When the user presses the Capture Key, a thread is created with a copy of the queue containing the raw frame data. This raw data is saved to a temporary file on disk.	FFmpeg is launched to convert the raw video file to the specified output formats (MP4 and/or GIF). After file output, the temporary file is deleted.

### What is FFmpeg and why is it included with Flashback Video Recorder?

FFmpeg is an open source, cross platform tool that allows users to capture, convert, and stream audio and video files. Flashback Video Recorder uses FFmpeg to convert raw frame data to the supported formats: GIF and MP4.

Neither Flashback Video Recorder, nor its creator, LaunchPoint Games, is associated with FFmpeg in any way. You are free to replace the included FFmpeg binaries with your own version should you choose.

You can learn more about FFmpeg and access the source code here:

<http://ffmpeg.org> and <http://www.ffmpegmac.net/>

### How can I improve performance when using Flashback Video Recorder?

There are two areas of performance that may be impacted by the Flashback Video Recorder: framerate and memory.

In our testing, having the Flashback Recorder enabled could add between 3 to 6 ms per frame, depending on hardware. To reduce this impact, you can lower the framerate or the capture resolution.

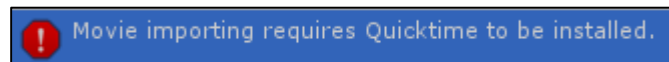
Saving raw video can also have a direct impact on memory usage. A video captured for 6 seconds at 30 frames a second with a resolution of 1280x720 can consume as much as 474.6 MB of memory. To reduce memory usage, select a lower resolution, reduce the framerate setting, and decrease the capture time.

To calculate memory usage in bytes, you can use the following equation:

$$\text{Mem\_Usage} = \text{frame\_height} * \text{frame\_width} * 3 * \text{framerate} * \text{capture\_time}$$

### Why do I get a QuickTime error when importing MP4 files?

You may get the following error if QuickTime is not installed on your Windows PC and you import MP4 videos in the Project view:



This is to be expected, and the error is harmless. For example, double clicking on an MP4 video in the Project View will launch it with the default player on your PC.

You may choose to install QuickTime, but be aware that Apple no longer officially supports the Windows version. You can find a link to QuickTime, along with Apple's statement, here: <https://support.apple.com/kb/DL837>

Also, keep in mind that this is a Unity3D error, and is not specific to Flashback Video Recorder. It will be up to Unity to implement another way to handle MP4 videos in the Editor.

### Why does my captured video look jittery?

There are several factors that impact the smoothness of the video playback. First, if the framerate of your game drops below the capture framerate, the resulting video will appear to stutter. You will either need to choose a lower capture framerate, reduce the size of the captured video resolution, and/or find other optimizations in your project to ensure a high(er) framerate.

Another factor to take into consideration is that Flashback Video Recorder renders additional frames several times a second. Adding a motion blur image effect will help smooth the transitions between frames, reducing jerkiness.

### How can I reduce the file size of my captured GIFs?

GIFs, by their very nature, are inefficient. The same capture might result in a 200kb MP4 video and a 30 MB GIF. It is strongly recommended that you choose a lower capture framerate (ex: 15 per second), a lower capture resolution (ex: 240x180), and a lower capture history (ex: 5 seconds) to reduce file size.

### Why do my GIFs appear low quality?

Each frame in a GIF can only store a maximum of 256 colors. This means that GIFs tend to be lower quality, or have artifacts, that are not present in either raw video or MPEG-4 video.

Video games which have many color shifts or a moving camera can exacerbate the problem.

### Can I capture more than 30 frames per second?

We chose the upper limit of 30 frames per second for two main reasons:

First, most video displayed at over 30 frames per second tends to look “cheap” because the motion is too smooth (think of the visual style in Peter Jackson’s Hobbit movies).

Second, most desktop games tend to have a target framerate of 60 frames per second or higher. Increasing the capture framerate requires more processing time and memory.

Since most video is between 20 and 30 frames per second, the additional resource overhead for a higher framerate did not seem to add much benefit.

That being said, there are two different ways to increase the capture framerate, both involve making changes in code.

Option 1: In the FlashbackRecorder.cs file, change the slider range values:

```
//Changes the capture framerate to a maximum value of 60
// frames per second
[SerializeField, Range(1, 60)]
int m_FrameRate = 15;
```

Option 2: You can set the capture framerate programmatically, which is not limited by the variable range:

```
//Records the last 8 seconds at 60 frames per second
FlashbackRecorder rec = GetComponent<FlashbackRecorder> ();
rec.UpdateCaptureSettings (8, 60);
```

### Why is video capture not working?

There are several things to try when troubleshooting video capture. Skip to Step 6 if the Flashback Video Recorder works in the editor, but not in your standalone build.

Step 1: Make sure the Flashback Video Recorder is attached to a Camera object and ensure it is enabled (the checkbox next to the name of the component is ticked).

Step 2: Make sure a “Capture Key” is selected, or if it is set to “None”, then make sure you are actually calling “SaveCapturedFrames”.

Step 3: Ensure FFmpeg is in the correct folder. When running in the Unity Editor, FFmpeg must be in the /Assets/FlashbackRecorder/FFmpeg/ directory. If you are running on a Windows machine, you will need the “FFmpegWin.exe” in that folder, and if you’re running on a Mac, then you’ll need the “FFmpegMac” binary in that folder.

Step 4: Ensure you are looking in the correct output directory. You will need to verify that your game has permission to read, write, delete, and create files in the specified output folder.

Step 5: If running in the editor, make sure you right click in the “Project” view and select “Refresh” since the Asset manager might not import the generated files automatically.

Step 6: If running standalone, make sure you click one of the “Prepare Standalone Build” buttons (for the appropriate platform) in the component window. This will write the FFmpeg binary to the /StreamingAssets/FlashbackRecorder/ folder, which is required for Flashback Recorder to work in the final build.

## Why do shadows disappear when I'm using Flashback Video Recorder?

Flashback Video Recorder uses video memory to store frames during capture. Since this is a shared resource, Unity will dynamically allocate shadow texture memory based on the total memory available. In other words, Unity will reduce shadow quality, then remove them entirely, if Flashback Video Recorder is using too much video RAM.

You can learn more here: <https://docs.unity3d.com/Manual/LightPerformance.html>

To mitigate this problem (short of upgrading your video card), you can do the following:

1. Reduce capture resolution. Cutting the resolution in half will cut RAM use by 4x.
2. Reduce capture time. The more video that needs to be recorded, the more GPU memory is required.
3. Reduce capture framerate. The fewer frames that need to be captured, the less video memory is required.

## What if I have improvements or new features for Flashback Video Recorder?

If you have modified Flashback Video Recorder with a new feature or improvement (performance, quality, etc.), then we want to hear from you in order to give proper credit! Just send an email to [support@launchpointgames.com](mailto:support@launchpointgames.com) and point out that you have made a change to Flashback Video Recorder. Include a brief description of what you have added or changed, as well as why you think its valuable, and someone will get in touch shortly.

If your update is non-trivial and/or teaches us something new, we want to say thank you. We would even consider giving a reward, such as a full or partial refund, depending on the size, complexity, and time it took to make the improvement.

## 7. Features Roadmap

Flashback Video Player is far from perfect, and we have identified several areas of improvement and feature sets we want to add. The list below is currently what is on the roadmap, but it is subject to change without notice. Additionally, we cannot commit to any specific timeframe or dates.

And to be clear, all new features and product updates will be added to the Flashback Video Recorder .unitypackage in the Unity Store and will be free of charge.

Feature	Description	Priority
Add Watermark	Optionally add a watermark to video captured in your game (for example, your company logo or game name).	High
Long video capture	Currently Flashback Video Recorder works well for videos of 1 minute or less. We want to support longer videos of indefinite length.	Medium
In-Game Playback	In addition to capturing video in your game, we want to implement a method for playing it back.	Low
Mobile Platforms	Support for iOS and Android devices to bring them on par with the experience on desktop.	Low

## 8. Terms & Conditions

These terms and conditions outline the rules and regulations for the use of Flashback Video Recorder. You must purchase Flashback Video Recorder from the Unity Asset Store, or have written permission from LaunchPoint Games, in order to obtain a license to use Flashback Video Recorder in your personal or commercial project. One license is required per seat.

Unless otherwise stated, LaunchPoint Games and/or its licensors own the intellectual property rights for all material found in Flashback Video Recorder.

Flashback Video Recorder contains FFmpeg, which is covered by the [GNU Lesser General Public License \(LGPL\) version 2.1](#). Full legal terms can be found on the FFmpeg website along with the source code.

FFmpeg License and Legal Considerations: <http://ffmpeg.org/legal.html>  
FFmpeg Download and Source: <http://ffmpeg.org/download.html#get-sources>

You may use and modify the Flashback Video Recorder source code for use in your projects. However, you may not distribute the package, or any components included with the package covered by this agreement, without written consent from LaunchPoint Games. Flashback Video Recorder is provided “as is” and all other intellectual property rights are reserved.

LaunchPoint Games is not responsible for any damages as a result of using Flashback Video Recorder, including, but not limited to, loss of productivity, loss of data, or legal fees or damages.

If you do not agree to the above terms and conditions, you must immediately stop using Flashback Video Recorder, remove any and all components deployed by the Flashback Video Recorder asset package from your projects, and delete the asset package from your local storage devices, remote storage devices, and version control software.