Readme - Pro GIF 1.8.1

Pro GIF is a mega-package of GIF features. Handles most of the GIF file processing and provides very efficient ways for the integration and management of GIF features. Get resolve to tons of problems for related developments. The Pro GIF encoder and decoder are highly optimized and enhanced. Run in threads for better performance and support multiple instances. Super fast, stable, powerful and flexible to integrate!
So many useful stuff includes in this package for use in different cases, make this package a powerful toolkit for creating GIF features for your game and application!

Highlights

- **The Ultimate GIF playback and recording solutions.**
- **Pro GIF Advanced Recorder & Decoder.**
- **Ultra-low playback memory footprint. Even for large number of GIF frames decoded and stored in the memory.**
- **Decoder Version 2. Up to 9 times faster than decoder version 1.**
- **Super fast decoder (V2), great in performance and compatibility. Decode GIF simlilar to Chrome, Firefox browser, etc. Plus great functionality gives you the best flexibility for handling GIF.**
- **Both the encoder and decoder support run in threads**, which means they will still work in the background if the users switch to other apps, watching a video Ad, or be redirected to a web page by clicking on a banner Ad in your app. You can show the GIFs when they come back. Have a try: GooglePlay, AppStore

Index

✔ Include the Pro GIF assets to 3rd parties:

You are allowed to include the Pro GIF assets in your project/solution/work to 3rd parties for the following condition(s):

Only if the 3rd parties you work for/with, have purchased the same or greater edition of the Pro GIF assets.

(1) Features

<Core>

- Record GIF/Convert still images to GIF(support transparent).
- Record GIF with camera(s).
- Play GIF(support Transparent, Variable Frame Rate, Reverse & Ping-pong play mode).
- Load GIF file from local-path/URL, decode, and playback. With option for saving file in local storage.
- Rich settings: FPS, Duration, Quality, Repeat Count, Aspect Ratio, Transparent Color(for hiding a particular BG color), Resolution(support auto resize to fit any screen size).
- Enable/Disable auto-detection of image transparent setting for recorder.
- The encoding & decoding process runs in thread for better performance.

<Advanced>

- Advanced decode settings, allows setting how many frames to decode.
- Ultra-low memory footprint, even for large number of gif frames.
- Supports multiple GIF decode and playback.
- Supports multiple GIF recorders with different cameras.
- Convert JPG/PNG/Texture2D (List) to GIF.
- Crop GIF (with a specific aspect ratio, e.g. 16:9, 3:2, 4:3, 1:1, etc.).
- Rotate GIF (90, -90, 180 degrees).
- Support save Reverse and Ping-pong play mode GIF.
- Easily get the GIF info: image size, frame count, fps, and the first frame texture.
- Supports display gifs on Image, RawImage, Renderers(Meshes such as Cube, Plane, Sphere etc...), GuiTexture and any other material that support Texture2D, Sprite, or RenderTexture.
- Provides 2 GIF Managers: flexible API, ready to use, auto memory management, fully tested.

<Extra>

- Mobile Media Plugin for saving and picking images (including animated GIF) from Android Gallery and iOS Photos, and more.
- API helper classes for easily use the GIF API to access the world's largest GIF library.
- Extra API helper classes: Weather, TimeZone & Giphy etc.
- Use your own GIF channels & API keys (Giphy).
- Share on up-to 15 social platforms.
- Optimized Json tool(Newtonsoft.Json), work on mobile & desktop.
- OnEditorGifRecorder for recording gif in the Editor play mode. Record the development screens at any time for your app/game promotion on social platforms.
- Some more useful stuff.

Support: Android, iOS, Windows, Mac, Linux, Unity Editor.

(2) Reminders, Setup & Requirement

Build iOS: **.NET 2.0** is required for Newtonsoft.Json to work properly on iOS, please select **.NET 2.0** in Player Settings before building XCode project. **(File > Build Settings > Player Settings > Other Settings > Optimization > Api Compatibility Level)**

Setup MobileMedia plugin
Requires Android 4.3(API Level 18) or later for Android platform,
Requires iOS 8.0 or later for iOS platform.
For Android, set **Write Permission** to "External (SDCard)" in Unity "Player Settings".

GUITexture (obsolete)
This component is obsolete, so we adds a define symbol for it: **PRO_GIF_GUITEXTURE**
All the GUITexture related scripts in this asset will be hidden by default. But you can re-enable them by insert the define symbol in the Unity editor(***Player Settings > Scripting Define Symbols***).

Skip Optional GIF Extensions
You can skip some optional extensions(PlainText & Comment) in GIFs. Skipping these extensions may give the decoder better compatibility with more GIFs (some rare GIFs that have been customized in a little bit different way in the extension fields).
Set the global flag to enable/disable the optional extensions (true: skip; false: don't skip):

```
ProGifDecoder.SkipOptionalExtensions = true;
```

*** Please noted that all the availability of the APIs helper classes provided with this asset may depend on related 3rd party services. We can't guarantee all the 3rd party services' availability, quality, as time goes. However, we will do our best to maintain the asset!

(3) PGif : Multiple Recorders (Encoder)

The **PGif** manager is recommended if you have multiple cameras for recording GIFs in the scene. The easiest way to record GIFs with multiple cameras. It is very simple as belows:

Record multiple GIFs using PGif:
PGif.iStartRecord(Camera:camera, string:RecorderName);

* For multiple recorder use case, just specify a unique name for each recorder.

Use that unique name to access and control the recorder(s):
PGif.iPauseRecord(string:RecorderName);
PGif.iResumeRecord(string:RecorderName);
PGif.iStopRecord(string:RecorderName);
PGif.iSaveRecord(string:RecorderName, string:optionalGifFilename);
PGif.iClearRecord(string:RecorderName);

Customize settings for recorder(encoder), call the below method before recorder start:
PGif.iSetRecordSettings(bool:autoAspect, int:width, int:height,
float:duration, int:fps, int:loop, int:quality);

PGif.iSetRecordSettings(Vector2:aspectRatio, int:width, int:height,
float:duration, int:fps, int:loop, int:quality);

Sets the GIF rotation
PGif.iSetGifRotation(ImageRotator.Rotation: rotation);

Sets the GIF transparent color (hide this color in the GIF)
PGif.iSetTransparent(Color32: color);

Auto detect transparent color (for prepared images those have tranparent background)
PGif.iSetTransparent(bool: autoDetectTransparent);

Set the GIF play mode before Save (Normal, Reverse, Ping-pong)
PGif.iGetRecorder(string: recorderName).recorderCom.m_EncodePlayMode =
ProGifRecorderComponent.EncodePlayMode.Reverse;

Callbacks

These callbacks are automatically handled by the GIF managers(PGif /ProGifManager).
Just assign your methods/Actions for receiving updates from them.
OnRecordProgress
OnRecordDurationMax
OnPreProcessingDone;
OnFileSaveProgress;
OnFileSaved;

More parameters and methods available in PGif class or through the iGetRecorder and iGetPlayer method in the class.

(4) PGif : Multiple Players (Decoder)

Decoder Introduction

The ProGif decoders support decode multiple GIFs at the same time. There are 2 decoder options and 2 decode modes.

- **Two decoder options**(Coroutine and Thread): we have introduced the thread decode option since v1.5.0. The thread solution has better performance than coroutines for playing few gifs at the same time, while the coroutines solution can maximize the use of device computation power when the number of gifs increases. No matter you are using the thread or coroutine option, they are enough fast for most of the use cases(eg. chatroom gif stickers, showing Giphy/Tenor preview version of gifs in the endless scroll list).
  The main point for using the thread option is it does not block the main thread, so your app can remain smooth(recommended for scrollable views). It also supports decode in background, means you can start decode some GIFs at a time, and then press the home button of the device. The decode process will go on until finish all decode task.

- **Two decode modes**(Normal, Advanced): the Normal decode mode decode the entire GIF normally, the Advanced decode mode allows setting the number of frames to decode.

- **Optimize Memory Usage** option: use a little computing resource as a trade-off for saving memory usage for storing GIF textures. This option is enabled by default, you can turn it on or off as you need. Set it through the provided GIF managers(PGif/ProGifManager). Set the OptimizeMemoryUsage flag before decode/play a GIF.

The **PGif** manager is recommended for displaying multiple GIFs at a time.
Pro GIF player also supports loading gif from local path or web url, and decode the gif to Textures/Sprites/RawTextures for playing in the scene.

Play multiple GIFs using PGif:

```
PGif.iPlayGif(ProGifRecorder:recorderSource, Image:playerImage,
        string:playerName, Action<float>:onLoading);
```

* For the display target, supports Image, RawImage, Renderers(Meshes such as Plane, Cube, Sphere, etc, and GuiTexture).
* For multiple player use case, just specify a unique name for each player.

Use that name to access and control the player(s):

```
PGif.iPausePlayer(string:PlayerName);
PGif.iResumePlayer(string:PlayerName);
PGif.iStopPlayer(string:PlayerName);
PGif.iClearPlayer(string:PlayerName);
```

Set the flag to enable/disable Memory Usage Optimization:

```
PGif.iSetPlayerOptimization(bool:enable);
```

Set Ping-pong play mode:
PGif.iGetPlayer(string:PlayerName).PingPong();

Set Reverse play mode:
PGif.iGetPlayer(string:PlayerName).Reverse();

Customize settings for player(decoder), call the below method before play gif:
PGif.iSetAdvancedPlayerDecodeSettings(decoderOption, targetDecodeFrameNum,
framePickingMethod, framesToDecode, optimizeMemoryUsage);

Get a target GIF info: decode the first frame to get detailed info(first frame texture, width, height, fps, total frame count, interval):
PGif.iGetGifInfo(string: gifPath, Action<ProGifPlayerComponent.FirstGifFrame>: onComplete,
ProGifPlayerComponent.Decoder: decoder);

Callbacks

Reports the loading progress, instantly finish if play GIF using the recorder source.
PGif.iGetPlayer(string:PlayerName).SetLoadingCallback(Action<float>: onLoadingCallback);

The callback to be fired on every frame during play GIF. Pass a GifTexture each time.
PGif.iGetPlayer(string:PlayerName).SetOnPlayingCallback(
Action<GifTexture>: onPlayingCallback);

The callback to be fired when the first gif frame ready.
PGif.iGetPlayer(string:PlayerName).SetOnFirstFrameCallback(
Action<FirstGifFrame>: onFirstFrameCallback);

The callback to be fired when all frames decode complete.
PGif.iGetPlayer(string:PlayerName).SetOnDecodeCompleteCallback(
Action<DecodedResult>: onDecodeCompleteCallback);

More parameters and methods available in PGif class or through the iGetRecorder/iGetPlayer method in the class.

(5) PGif : Clean Up Memory

The GIF Manager handles memory clean up when a recorder or player restart, but in case you want to implement the Clear methods manually. You can use the below methods:

Clear the target recorder by recorder name:
PGif.iClearRecorder(string:recorderName);

Clear the target recorder and ensure the textures not being cleared too early(for the case the recorder source is in use by a gif player):
PGif.iClearRecord_Delay(string:RecorderName, string:playerName, Action<string>:onClear);

Clear the target player by player name:
PGif.iClearPlayer(string:playerName);

(6) OnEditorGifRecorder

The OnEditorGifRecorder is an editor script with prefab, for recording gif in the Unity editor play mode. Useful to record the development screens at any time.

How to use?
-Drop the prefab(**OnEditorGifRecorder**.prefab) to the scene,
-Make some setting changes in the inspector if need,
-Click the 'Start Record' button to start the GIF recorder,
-Click the 'Save Record' button to save the stored frames as GIF.
-Wait the save progress to finish.

(7) ProGifManager : Recorder (Encoder)

To setup and start
Get/Create an instance of ProGifManager:
ProGifManager gifMgr = ProGifManager.Instance;

Call the methods like this:
gifMgr.MethodName(…);
or
ProGifManager.Instance.MethodName(…);

To make changes to the recorder settings:
ProGifManager.Instance.SetRecordSettings(bool: autoAspect, int: width,
int: height, float: duration, int: fps, int: repeatCount, int: quality);
Or
ProGifManager.Instance.SetRecordSettings(Vector2: aspectRatio, int: width,
int: height, float: duration, int: fps, int: repeatCount, int: quality);

Start gif recording (Camera.main will be used):
ProGifManager.Instance.StartRecord();
Or
ProGifManager.Instance.StartRecord(Action<float>: onRecordProgressCallback,
Action: onRecordDurationMaxCallback);

Start gif recording with specific camera:
ProGifManager.Instance.StartRecord(Camera: camera,
Action<float>: onRecordProgressCallback, Action: onRecordDurationMaxCallback);

To pause
Pause gif recording:
ProGifManager.Instance.PauseRecord();

To resume
Resume gif recording:
ProGifManager.Instance.ResumeRecord();

To stop
Stop gif recording, cannot be resumed, waiting to be saved/cleared:
ProGifManager.Instance.StopRecord();

To save stored frames to a gif file
ProGifManager.Instance.SaveRecord(string: optionalGifFilename);
Or
ProGifManager.Instance.SaveRecord(Action: onRecorderPreProcessingDoneCallback,
Action<int, float>: onFileSaveProgressCallback,
Action<int, string>: onFileSavedCallback, string: optionalGifFilename);

To stop and save stored frames to a gif file
Stop and save the recording:
ProGifManager.Instance.StopAndSaveRecord();
Or

```
ProGifManager.Instance.StopAndSaveRecord(Action: onRecorderPreProcessingDoneCallback,
        Action<int, float>: onFileSaveProgressCallback,
        Action<int, string>: onFileSavedCallback, string: optionalGifFilename);
```

Sets the GIF rotation

```
ProGifManager.Instance.SetGifRotation(ImageRotator.Rotation: rotation);
```

Sets the GIF transparent color (hide this color in the GIF)

```
ProGifManager.Instance.SetTransparent(Color32: color);
```

Auto detect transparent color (for prepared images those have tranparent background)

```
ProGifManager.Instance.SetTransparent(bool: autoDetectTransparent);
```

Set the GIF play mode before Save (Normal, Reverse, Ping-pong)

```
ProGifManager.Instance.m_GifRecorder.recorderCom.m_EncodePlayMode =
        ProGifRecorderComponent.EncodePlayMode.Reverse;
```

Callbacks

These callbacks are automatically handled by the GIF managers(PGif / ProGifManager).
Just assign your methods/Actions for receiving updates from them.

```
OnRecordProgress
OnRecordDurationMax
OnPreProcessingDone;
OnFileSaveProgress;
OnFileSaved;
```

(8) ProGifManager : Player (Decoder)

Decoder Introduction

- Same as **(4) PGif : Multiple Players (Decoder)**

To play gif after recording complete
Play the recorded gif frames stored in recorder:
ProGifManager.Instance.PlayGif(Image: targetImage, Action<float>:onLoading);

To play gif with filePath or Url
Load and decode a gif file and play it:
ProGifManager.Instance.PlayGif(string: gifPath, Image: targetImage,
Action<float>: onLoading, bool: shouldSaveFromWeb);

To pause / resume / stop gif player when a GIF is playing
Pause the player, the player will be paused at current frame:
ProGifManager.Instance.PausePlayer();

Resume the player, continue to play from current frame:
ProGifManager.Instance.ResumePlayer();

Stop the player, the player will be stopped and reset to first frame:
ProGifManager.Instance.StopPlayer();

Set the flag to enable/disable Memory Usage Optimization:
ProGifManager.Instance.SetPlayerOptimization(bool:enable);

Set Ping-pong play mode:
ProGifManager.Instance.m_GifRecorder.PingPong();

Set Reverse play mode:
ProGifManager.Instance.m_GifPlayer.Reverse();

Callbacks

Reports the loading progress, instantly finish if play GIF using the recorder source.
ProGifManager.Instance.SetPlayerOnLoading(Action<float>: onLoadingCallback);

The callback to be fired on every frame during play GIF. Pass a GifTexture each time.
ProGifManager.Instance.SetPlayerOnPlaying(Action<GifTexture>: onPlayingCallback);

The callback to be fired when the first gif frame ready.
ProGifManager.Instance.SetOnFirstFrameCallback(
Action<FirstGifFrame>: onFirstFrameCallback);

The callback to be fired when all frames decode complete.
ProGifManager.Instance.SetOnDecodeCompleteCallback(
Action<DecodedResult>: onDecodeCompleteCallback);

(9) ProGifManager : Clean Up Memory

The GIF Manager handles memory clean up when the recorder or player restart, but in case you want to implement the Clear methods manually. You can use the below methods:

Clear the recorder and player:
ProGifManager.Instance.Clear();
Or
Clear the recorder:
ProGifManager.Instance.ClearRecorder();

Clear the recorder and ensure the textures not being cleared too early(for the case the recorder source is in use by the player):
ProGifManager.Instance.ClearRecord_Delay(Action<string>:onClear);

Clear the player:
ProGifManager.Instance.ClearPlayer();

More parameters and methods available in ProGifManager class or through the m_Recorder and m_GifPlayer variable in the class.

(10) JPG/PNG/Textures to GIF

Use **ProGifTexturesToGIF** to convert images/textures to animated GIF. For examples:

1) Load **JPG** and **PNG** from specific directory in the application paths;
2) Import images from the Gallery to your application;

and, load and prepare the images as a texture list of **Texture2D**/**RenderTexture**. Converts and saves the textures as GIF with highly customizable settings!
Simple API, convenient, powerful features! (Demo scene: **TexturesToGIF_Demo**)

Converts/Saves textures to GIF:
It is very simple. Just prepare your textures in a texture list.
Call the Save method of ProGifTexturesToGIF. That's it!

```
ProGifTexturesToGIF.Instance.Save(....);
```

Or, create multiple ProGifTexturesToGIF instances by the Create method:

```
ProGifTexturesToGIF  texToGif = ProGifTexturesToGIF.Create(....);
texToGif.Save(....);
```

Sets the GIF rotation:

```
texToGif.SetGifRotation(ImageRotator.Rotation: rotation);
```

Sets the GIF transparent color (hide this color in the GIF)

```
texToGif.SetTransparent(Color32: bgColor);
```

Auto detect transparent color (for prepared images those have tranparent background)

```
texToGif.SetTransparent(bool: autoDetectTransparent);
```

If you have already imported images in the app accessible paths, you can use the **LoadImages** method to load images(JPG/PNG) from that path:

```
texToGif.LoadImages(string: inAppDirectory);
```

You may want to set the **image format** for loading into the texture list before LoadImages:

```
texToGif.SetFileExtension(List<string>: fileExtensions);
```

Customizable Settings:

width - Target width for the GIF.
height - Target height for thr GIF.
fps - Frame count per second.
frameDelay - Frame delay time in seconds.
loop - Repeat time, -1: no repeat, 0: infinite, >0: repeat count.
quality - (1 - 100) 1: best(larger storage size), 100: faster(smaller storage size).
resolutionHandle - The method to resize the textures if necessary.
autoClear - Auto clear all textures when the GIF is saved.
destroyOriginTexture - Clear the original textures after processed.
smooth_yieldPerFrame - Avoid blocking the main thread, so the UI smoother.

More parameters and functions available in ProGifTexturesToGIF class.

(11) Weather & TimeZone API Helper

To use World Weather Online API, please apply your own API keys here:
https://developer.worldweatheronline.com/api/
How to USE? Run the demo scene for details!
Demo scene included: **WWO-ApiDemo.unity**

(12) Crypto Currency API Helper

API Documentation: https://coinmarketcap.com/api/
How to USE? Run the demo scene for details!
Demo scene included: **CoinMarketCapApp.unity**

(13) Giphy API Helper

To use Giphy API, it requires a Giphy account to create API KEY and Upload API Key.
You need to request a production key for the Upload API as well.

APPLY HERE: https://developers.giphy.com/dashboard

How to USE? Run the demo scene for details!
Demo scene included:

1. **GifApi+ProGifPlayer Demo.unity,**
2. **GifApiDemo.unity**

(14) Social Share

Share GIF/image Url(s) return by the Giphy APIs. GIF preview and playback depends on the social platform. Support up to 15 social platforms (Facebook, Twitter, Tumblr, VK, Pinterest, LinkedIn, Odnoklassniki, Reddit, GooglePlus, QQZone, Weibo, Baidu, MySpace, LineMe, Skype).

Share GIF and/or text message:

```
GifSocialShare gifShare = new GifSocialShare();

gifShare.ShareTo(Social: socialPlatformType, string: title, string: description,
        string: url1, string: url2);
```

(15) Mobile Media Plugin

The native plugin we have developed for mobile developers to save media and pick media from Android Gallery & iOS Photos.
Supported media type for both Android & iOS: still image, gif, video.

Requirements & Setup

Requires Android 4.3(API Level 18) or later for Android platform,
Requires iOS 8.0 or later for iOS platform.

For Android, set **Write Permission** to "External (SDCard)" in Unity3D "Player Settings".

Save media to native (Android & iOS):

- *Save the byte array(in memory) of a media:*
  MobileMedia.SaveBytes(byte[]: mediaBytes, string: folderName, string: fileName, string: extensionName, bool: isImage);

- *Copy an existing media from source path to destination path:*
  MobileMedia.CopyMedia(string: existingMediaPath, string: folderName, string: fileName, string: extensionName, bool: isImage);

- *Save a Texture2D as static JPG/PNG:*
  MobileMedia.SaveImage(Texture2D: texture2d, string: folderName, string: fileName, ImageFormat: imageFormat, int: quality);

- *Save the byte array of a video:*
  MobileMedia.SaveVideo(byte[]: mediaBytes, string: folderName, string: fileName, string: extensionName);

Android native media picker:

- *Pick image(eg. JPG, PNG, GIF) or video from Android Gallery:*
  MobileMedia.PickImage(Action<string>: onReceived, string: title, string: androidMimeType, bool: iOS_UsePopup, string: iOS_TempImageNameWithoutExtension)

  MobileMedia.PickVideo(Action<string>: onReceived, string: title, string: androidMimeType, bool: iOS_UsePopup);

iOS native media picker:

- *Pick image(eg. JPG, PNG, GIF) or video from iOS Photos:*
  MobileMedia.PickImageIOS(Action<string>: onReceived, bool: iOS_UsePopup, string: iOS_TempImageNameWithoutExtension);

  MobileMedia.PickVideoIOS(Action<string>: onReceived, bool: iOS_UsePopup);

- *Get iOS media file thumbnail in the album:*
  GetMediaPreviewPhoto_IOS(Action<string>: onReceived, int: mediaType, int: mediaIndex, int: targetSize, string: iOS_TempImageNameWithoutExtension);

(16) Demo Scenes

SimpleStartRecordDemo.unity
This scene shows the simplest steps to start, change settings and stop & save a GIF recording.

ProGifDemo_Panels_HideUI.unity
This scene shows the steps of record, playback and change settings with our UI templates. Check the canvas settings that allow this example record GIF without UI.

ProGifDemo_Panels_ShowUI.unity
This scene shows the steps of record, playback and change settings with our UI templates. Check the canvas settings that allow this example record GIF including UI.

ProGifDemo_SpecificCamera.unity
This scene demostrates the ability to record GIF with specific camera.

GifApiDemo.unity
This scene shows how to use the APIs of Giphy.com to download and play multiple GIFs in a scrollview, also with social share buttons.

ProGifDemo_MultipleCamera.unity
This scene demonstrates how to record GIFs with multiple cameras using different GIF settings.

TexturesToGIF_Demo.unity
This scene demonstrates how to load and convert images(JPG, PNG) to GIF.

ProGifDemo_PlayerRenderer.unity
This scene demonstrates displaying GIF with different renderers(i.e. Cube, sphere, capsule, cylinder, plane) at a time.

GifApi+ProGifPlayer Demo.unity
This scene demonstrates the use of Pro GIF with Giphy APIs. Download and display multiple GIFs at a time.

ProGifDemo_MobileMedia+GIF.unity
This scene shows how to save and pick image(including animated GIF) from Android Gallery and iOS Photos. And show to load and play the picked image(animated GIF).

THANK YOU

Thank you for using this package!
For any question and bug report please contact us at swan.ob2@gmail.com.
Remember to rate this asset on the Asset Store. Your review is always appreciated, and very important to the development of this asset!

Review And Rating

Visit our asset page to find out more!
https://www.swanob2.com/assets

SWAN DEV