Readme - Pro GIF Universal 1.2.6

Pro GIF Universal is the most completed GIF library for Unity. Ultra-performance, handles all sorts of GIF files, and provides very efficient ways for the integration and management of GIF features for any indie and even commercial projects!

The Pro GIF Universal encoder and decoder are highly optimized and enhanced. Run in threads for better performance and support multiple instances. Our high-performance library lets you record and playback GIFs instantly, no more waiting!

All our codes and examples are carefully designed to provide a clean, easy-to-use package. The Unified APIs allowing you to integrate once, runs on all supported platforms. GIF has never been so easy with Pro GIF series!

Highlights

- The Ultimate GIF playback, recording and converter solutions.
- Pro GIF Advanced Recorder, Player and Converter, highly customizable settings.
- Super fast, multi-threaded encoder.
- Super fast decoder, great in performance and compatibility. Decode GIF similar to Chrome, Firefox browser, etc. Plus great functionality that provides the best flexibility for handling GIFs.
- Ultra-low playback memory footprint. Even for a large number of gif frames decoded and stored in the memory.
- Decoder **Version 3**. Around 15 times faster than V1 and around 50% faster than V2, or even faster as it is further optimized in our continuous updates.
  (Decoder speed: ProGIF V1=100%, ProGIF V2=1000%, Universal V3=1500%)
- Multi-threaded decoder, >200% efficiency for multiple GIFs (compare to Pro GIF).
- Fully supports interlaced GIFs.

Index

✔ Include the Pro GIF Universal assets to 3rd parties:

You are allowed to include the Pro GIF Universal 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).
- Instant Preview/Play the newly recorded GIF.
- 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.
- Super fast, multi-threaded encoder.
- Ultra-low memory footprint, even for large number of gif frames.
- Supports multiple GIF decode and playback.
- Supports multiple GIF recorders with different cameras.
- Advanced Converter: 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.
- Support adding and getting Comment-Extension(human readable metadata, e.g. image credit, description).
- 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.
- Easy to use GIF Manager: flexible API, auto memory management, battle-tested!

<Extra>

- API helper classes for easily use the GIF API to access the world s largest GIF library. Search for any kinds of GIFs you want!
- Use your own GIF channels & API keys (Giphy).
- Share on up-to 15 social platforms.
- Optimized Json tool(Newtonsoft.Json), work on mobile & desktop.
- The codeless Pro GIF recorder (editor extension) for recording GIF in both the Editor play mode and runtime App, without modifying your code, just drag and drop the script/prefab to your scene.
- Some more useful stuff.
- GIF libraries full source code.

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

(2) Reminders, Setup & Requirement

Build iOS: **.NET 2.0 or newer** is required for Newtonsoft.Json to work properly on iOS.

* Newtonsoft.Json is used with the API helper classes(i.e. Giphy API) in this asset.

Reminded! After importing the aaset, there will be Newtonsoft class related errors if the project does not contain a Newtonsoft plugin. Don't worry, please find our Newtonsoft plugin in the folder: ***Assets/SWAN Dev/Newtonsoft***. Double click to import the included Newtonsoft package. If your project already has a Newtonsoft plugin installed, do not import it.

* The GIF library(encoder & decoder) can be used independently and compatible with .NET 2.0 Subset too. (.NET 2.0 Subset and newer compatible)

Setup for Scriptable Render Pipeline (SRP) : URP/LWRP/HDRP

If your project is using Scriptable Render Pipeline (e.g. URP/LWRP/HDRP), in order to let the recorder to record the frames you have to insert the define symbol **PRO_GIF_SRP** in the Unity Editor(***File > Build Settings > Player Settings > Other Settings > Scripting Define Symbols***).

Skip Optional GIF Extensions

The below global variable allows the decoder to skip some optional extensions like PlainText and Comment in the GIF. (true: skip optional extensions; false: don't skip):

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

Try Decode Broken GIF

The below global variable allows the decoder to skip the error and try to decode the good frames in the GIF. (true: try decode broken GIF; false: don't decode):

```
ProGifDecoder.TryDecodeBrokenGif = true;
```

MobileMedia Plugin

The MobileMedia Plugin provides your apps the ability to save and pick media files to/from the Android and iOS device Gallery/Photos. Once you have the plugin installed, you may add the define symbol "**SDEV_MobileMedia**" in the Unity ***PlayerSettings > OtherSettings > Scripting Define Symbols***.

This define symbol enables the MobileMedia features for the CodelessProGifRecorder editor extension and some Pro GIF demo scenes.

Check it out: https://www.swanob2.com/mobile-media-plugin

Play GIFs on WebGL

Set the decoder to run in coroutine mode(WebGL does not support threads), by adding the below code before playing GIFs.

- For PGif manager:

```
PGif.iSetAdvancedPlayerDecodeSettings(ProGifPlayerComponent.Decoder.ProGif_Coroutines, ...);
```

- For GIF player components:

```
playerComponent.SetAdvancedDecodeSettings(ProGifPlayerComponent.Decoder.ProGif_Coroutines, ...);
```

(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 below:

Record multiple GIFs using PGif:
PGif.iStartRecord(Camera:camera, string:RecorderName, ...);
* For multiple recorder use case, please 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, byte:colorRange);

Auto detect transparent color (for prepared images those have transparent 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

The below callbacks are handled in the iStartRecord method, please register these callbacks if need, by assigning your methods/Actions to receive updates from them. Confuse? Don't worry, there are plenty of demo scenes included, we will show you step by step in the scenes.

Action<float>: onRecordProgress
Action: onRecordDurationMax
Action: onPreProcessingDone
Action<int, float>: onFileSaveProgress
Action<int, string>: 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, this method and its variants supports UI Image, UI RawImage, Renderers (Meshes such as Plane, Cube, Sphere, etc, and GuiTexture).
* For multiple player use case, please specify a unique name and display target 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);
```

Get the existing player by name:

```
ProGifPlayer proGifPlayer = PGif.iGetPlayer(string:PlayerName);
```

Then you can access more variables, methods, components in the proGifPlayer object, for making more complicated features if you want.

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

Get GIF info without decoding any frame:
WebRequest/WWW method:
PGif.GetGifInfo(string:gifPath, Action<ProGifDecoder.GifInfo>:onComplete);
or
System.IO method:
ProGifDecoder.GifInfo gifInfo = GetGifInfo(string:gifPath);

Callbacks

The callback to be called when the GIF header is obtained (before decode the gif frames).
PGif.iSetPlayerOnHeaderLoaded(Action<ProGifDecoder.GifInfo>:onHeaderLoaded);

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

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

The callback to be fired when the first gif frame ready.
PGif.iSetPlayerOnFirstFrame(string:PlayerName, Action<FirstGifFrame>:onFirstFrame);

The callback to be fired when each frame decode is finished.
PGif.iSetPlayerOnFrameLoaded(Action<GifTexture, int, float>: onFrameLoaded);

The callback to be fired when all frames decode complete.
PGif.iSetPlayerOnDecodeComplete(Action<DecodedResult>:onDecodeComplete);

The callback to be fired when the decoder(player) encountered an error during decoding.
PGif.iSetPlayerOnGifError(string:PlayerName, Action<string>:onGifError);

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(e.g. the recorder source is being imported to the GIF player to preview):
PGif.iClearRecord_Delay(string:RecorderName, string:playerName, Action<string>:onClear);

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

(6) CodelessProGifRecorder

This is a code-less GIF recording feature designed for recording GIFs without changing your codes. It provides a convenient way for recording GIFs in your app/game, with some simple setup in the scene. So you can create GIFs at any time when you need to.

How to use?

Use it in the Editor:
- Drop the prefab(**CodelessProGifRecorder**.prefab) to your scene,
- Run 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 for the save progress to finish.

Use it in your App at runtime:
- You can also reference the methods and dynamic parameters in the CodelessProGifRecorder to your UI components like Button, Slider, InputField, and Toggle, etc. in the scene, this allows you to record GIFs at runtime in your app.

(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>: onRecordProgress,
Action: onRecordDurationMax);

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

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: onRecorderPreProcessingDone, Action<int, float>: onFileSaveProgress, Action<int, string>: onFileSaved, 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: onRecorderPreProcessingDone,
Action<int, float>: onFileSaveProgress,
Action<int, string>: onFileSaved, 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, byte colorRange);

Auto detect transparent color (for prepared images those have transparent 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
The below callbacks are handled in the StartRecord, SaveRecord and StopAndSaveRecord methods, please register these callbacks if need, by assigning your methods/Actions to receive updates from them.
Action<float>: onRecordProgress
Action: onRecordDurationMax
Action: onPreProcessingDone
Action<int, float>: onFileSaveProgress
Action<int, string>: onFileSaved

Any confuse? Don't worry, there are plenty of demo scenes included, we will show you step by step in the scenes.

(8) ProGifManager : Player (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
The callback to be called when the GIF header is obtained (before decode the gif frames).
ProGifManager.Instance.SetPlayerOnHeaderLoaded(Action<ProGifDecoder.GifInfo>:onHeaderLoaded)

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

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

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

The callback to be fired when each frame decode is finished.
ProGifManager.Instance.SetPlayerOnFrameLoaded(Action<GifTexture, int, float>: onFrameLoaded)

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

The callback to be fired when the decoder(player) encountered an error during decoding.
ProGifManager.Instance.SetPlayerOnGifError(Action<string>:onGifError)

(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 local storage or by URL;
2) Screenshots captured in the game;

and, prepare the images as a list of Texture2D/RenderTexture. Convert/Save the textures to a GIF with the settings (FPS, Resolution, etc.) you want.
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, byte: colorRange);
```

Auto detect transparent color (for prepared images that have transparent 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);
```

And, you can set the **image format** for loading images, before calling **LoadImages** method:

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

Customizable Settings:

width - Target width for the GIF.
height - Target height for the 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) GIF Remake (GIF to GIF)

Use ProGifRemaker to load an existing GIF from local-path or by URL, re-save it to a new GIF with new settings. The remaker extension can be used in the editor inspector without coding, just add the ProGifRemaker.cs script to an empty GameObject and follow the tips in the inspector to remake your GIFs.

You can also integrate its APIs into your script to use in your apps at runtime.

1. Get/Create an instance of ProGifRemaker:

```
ProGifRemaker remaker = ProGifRemaker.Instance;
```

2. Change the settings in m_Settings object:

```
remaker.m_Settings.VariableName1 = .... ;
remaker.m_Settings.VariableName2 = .... ;
. . . . . .
```

3. Call the **Remake** method, that's it.

```
remaker.Remake(string:originGifUrl, Action<int, string>:onFileSaved,
                                    Action<int, float>:onFileSaveProgress);
```

(12) Giphy API Helper

To use Giphy API, it requires a Giphy account to apply for the API keys.
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

You can also find our API documentation on our site:
https://www.swanob2.com/jsontool

(13) 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, QQZone, Weibo, Baidu, MySpace, LineMe, Skype).

Share GIF and/or text message:

```
GifSocialShare gifShare = new GifSocialShare();

gifShare.ShareTo(Social: socialPlatform, string: title, string: description,
        string: url1, string: url2, long: phoneNo, string: tags)
```

(14) Demo Scenes

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

ProGifDemo_Panels_Show_or_Hide_UI.unity
This scene shows the steps of record, playback and change settings with our UI templates. Also showing how to record GIF with or without UI, by changing the UI Canvas render mode.

ProGifDemo_SpecificCamera.unity
This scene demonstrates 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).

ProGifDemo_TransparentSetupExample.unity
This scene shows how to create better quality transparent GIF, plus some tips in the scene.

ProGifDemo_3DSceneUGUI.unity
This scene shows how to play GIFs in a 3D scene using UGUI Image/RawImage components.

THANK YOU

Thank you for using our assets!
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