Readme - Game GIF 1.5.1

Game GIF is a powerful GIF recorder package for creating, saving, sharing GIF, and more. It is a lite version of Pro GIF so you can easily upgrade to get more features when you are ready to push your app on to the next level!
This package provides the features to record animated GIFs and playback the newly created GIF frames immediately. Advanced, powerful GIF recorder, highly customizable settings. So many useful stuff included that make this package not only a recorder but also a toolkit for making & handling GIFs for game & application!
All our code and examples are carefully designed to provide a clean, easy to use package. GIF has never been so easy!

Highlights

- The Ultimate GIF recording and preview playback solutions.
- Super fast, multi-threaded encoder.
- Ultra-low playback memory footprint.
- Save Reverse and Ping-pong play mode GIF.
- Advanced recorder features(Crop, Rotate, Transparent background GIF, Multiple recorder instance).

Index

(1) Features

<Core>

- Record GIF/GIF Replay (support transparent).
- Preview/Play the recorded GIF (support transparent, support Reverse and Ping-pong play mode).
- Rich settings: FPS, Duration, Quality, RepeatCount, Aspect Ratio, Resolution(support auto resize to fit any screen size), Transparent Color(for hiding a particular BG color).
- The encoding process runs in threads for better performance.

<Advanced>

- Support record GIF with multiple cameras.
- Multi-threaded encoder.
- Ultra-low (preview player) memory footprint, even for large number of GIF frames.
- 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 display gifs on Image, RawImage, Renderers(Meshes such as Cube, Plane, Sphere etc...), GuiTexture and any other material that support Texture2D, Sprite, or RenderTexture.

<Extra>

- Mobile Media Plugin for saving and picking images (including animated GIF) to/from Android Gallery and iOS Photos, and more.
- GIF API helper classes for uploading and getting uploaded GIFs from Giphy.
- Use your own GIF channels & API keys.
- 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.
- 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. It will not be included in your build if you don't use the API.
* The GIF library(recorder/encoder) can be used independently and compatible with .NET 2.0 Subset too. (.NET 2.0 Subset and newer compatible)

Setup MobileMedia plugin

Requires Android 4.4(API Level 19) 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".

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***).

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(***File > Build Settings > Player Settings > Other Settings > Scripting Define Symbols***).

*** Please noted that all the availability of the APIs helper classes provided with this asset may depend on related 3rd party services(e.g. Weather, Timezone & Giphy API, if provided). 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 below:

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);

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

Set the GIF transparent color (hide this color in the GIF)
PGif.iSetTransparent(Color32:color, byte colorRange);

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 (GIF Preview)

The ProGif preview player support playback of multiple GIF recorder sources at the same time (in case you have set up multiple recorders). Also support Optimize Memory Usage option to minimize the memory consumption.

The **PGif** manager is recommended for displaying multiple newly created GIFs at a time.

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();

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);

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

Set the GIF rotation
ProGifManager.Instance.SetGifRotation(ImageRotator.Rotation:rotation);

Set the GIF transparent color (hide this color in the GIF)
ProGifManager.Instance.SetTransparent(Color32:color, byte colorRange);

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 (GIF Preview)

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

* For the display target, supports Image, RawImage, Renderers(Meshes such as Plane, Cube, Sphere, etc, and GuiTexture).

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);

(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) 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

Create/Get the GIF API Manager instance:
```
GiphyManager giphyMgr = GiphyManager.Instance;
```

Call the methods like this:
```
giphyMgr.MethodName(…);
```
or
```
GiphyManager.Instance.MethodName(…);
```

Set your channel UserName and Keys:
```
GiphyManager.Instance.SetChannelAuthentication(string: userName,
        string: apiKey, string: uploadApiKey);
```

Upload GIF:
```
GiphyManager.Instance.Upload(string: gifFilePath, List<string>: tagList,
        Action<GiphyUpload.Response>: onCompleteAction,
        Action<string>: onProgressAction);
```

Get GIF By Id:
```
GiphyManager.Instance.GetById(string: giphyGifId,
        Action<GiphyGetById.Response>: onCompleteAction);
```

(11) 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)
```

(12) Mobile Media Plugin

The native plugin we have developed for mobile developers to save media and pick media file from Android Gallery and iOS Photos.

Requirements & Setup

Requires Android 4.4(API Level 19) 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 file to native (Android, iOS)**

  SaveBytes(byte[]: mediaBytes, string: folderName, string: fileName,
  string: extensionName, MediaType: mediaType);

  CopyMedia(string: existingMediaPath, string: folderName,
  string: fileName, string: extensionName, MediaType: mediaType);

  SaveImage(Texture2D: texture2d, string: folderName,
  string: fileName, ImageFormat: imageFormat, int: quality);

  SaveVideo(byte[]: mediaBytes, string: folderName,
  string: fileName, string: extensionName);

  * **MediaType** enum (Image, Video, Audio_Android)

- **Save Audio to native (Android Only)**

  SaveAudioAndroid(byte[]: mediaBytes, string: folderName,
  string: fileName, string: extensionName);

- **Pick Image, Video, GIF from native (Android, iOS)**

  PickImage(Action<string>: onReceived, string: title, string: androidMimeType,
  bool: iOS_UsePopup, string: iOS_TempFileName);

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

- **Pick Audio from native (Android Only)**

  PickAudioAndroid(Action<string>: onReceived, string: title,
  string: androidMimeType, bool: iOS_UsePopup);

- **Get Image, Video, GIF thumbnail and full-size image (Android, iOS)**

  GetMediaThumbnail(Action<string>: onReceived, int: mediaType,
  int: mediaIndex, int: targetSize, string: Android_TargetFolderName);

  GetMediaPhoto(Action<string>: onReceived, int: mediaType,
  int: mediaIndex, string: Android_TargetFolderName);

(13) Demo Scenes

SimpleStartRecordDemo.unity
This scene shows the simplest steps to start, change settings and stop/save a recording for a game.

ProGifDemo_Panels_HideUI.unity
This scene shows the steps of record, playback and change settings with our UI templates. Pay attention to the canvas setting that let this example exclude UI from camera.

ProGifDemo_Panels_ShowUI.unity
This scene shows the steps of record, playback and change settings with our UI templates. Pay attention to the canvas setting that let this example include UI from camera.

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

GifApiDemo_GetById.unity
This scene shows how to get GIF from Giphy.com by id. And, share the GIF with its Bitly_Url/Url/Id responded by the API.

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

MobileMediaTest.unity
This scene simply shows how to pick and save media files to Android Gallery & iOS Photos.

THANK YOU

Thank you for using this package!
For any problem 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