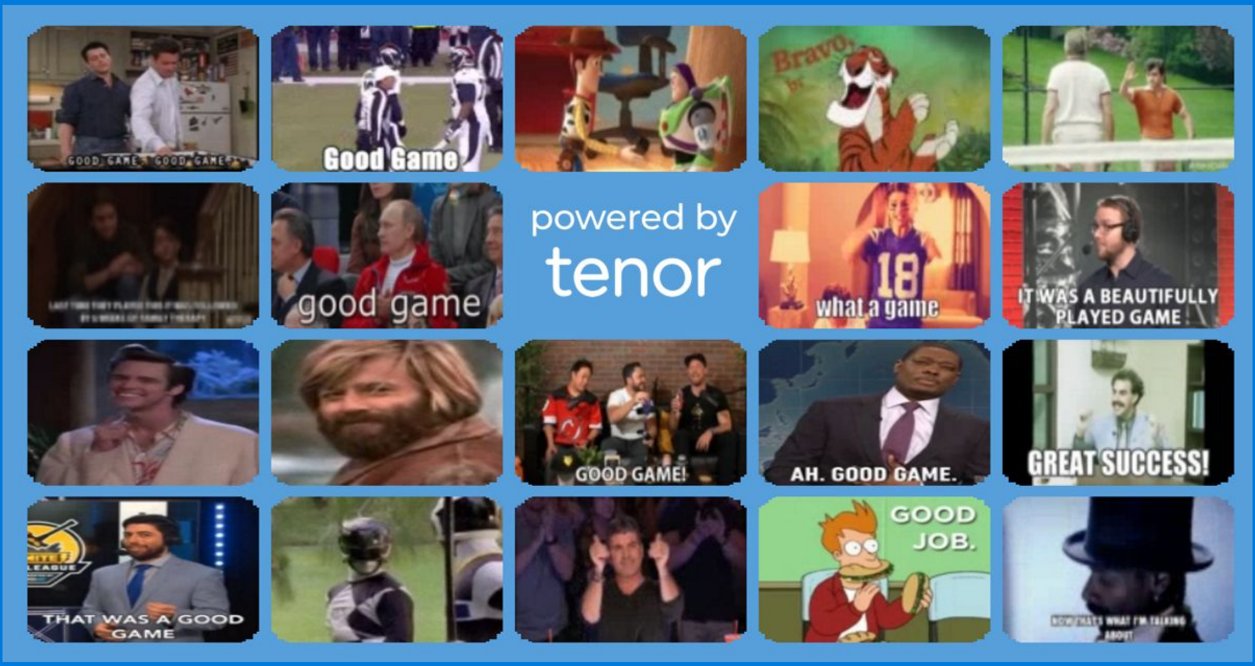
GIF Toolkit - powered by Tenor Unity Plugin

Documentation



Created and Written

By Harley Torrisi

http://harley-torrisi.me/
https://assetstore.unity.com/publishers/38764
https://harley-torrisi.itch.io/

Table of Contents

Introduction

With this plugin, you will be able to use Unity to access the Tenor API. Giving you the ability to search GIFs and display in-game through a Unity Video Player.

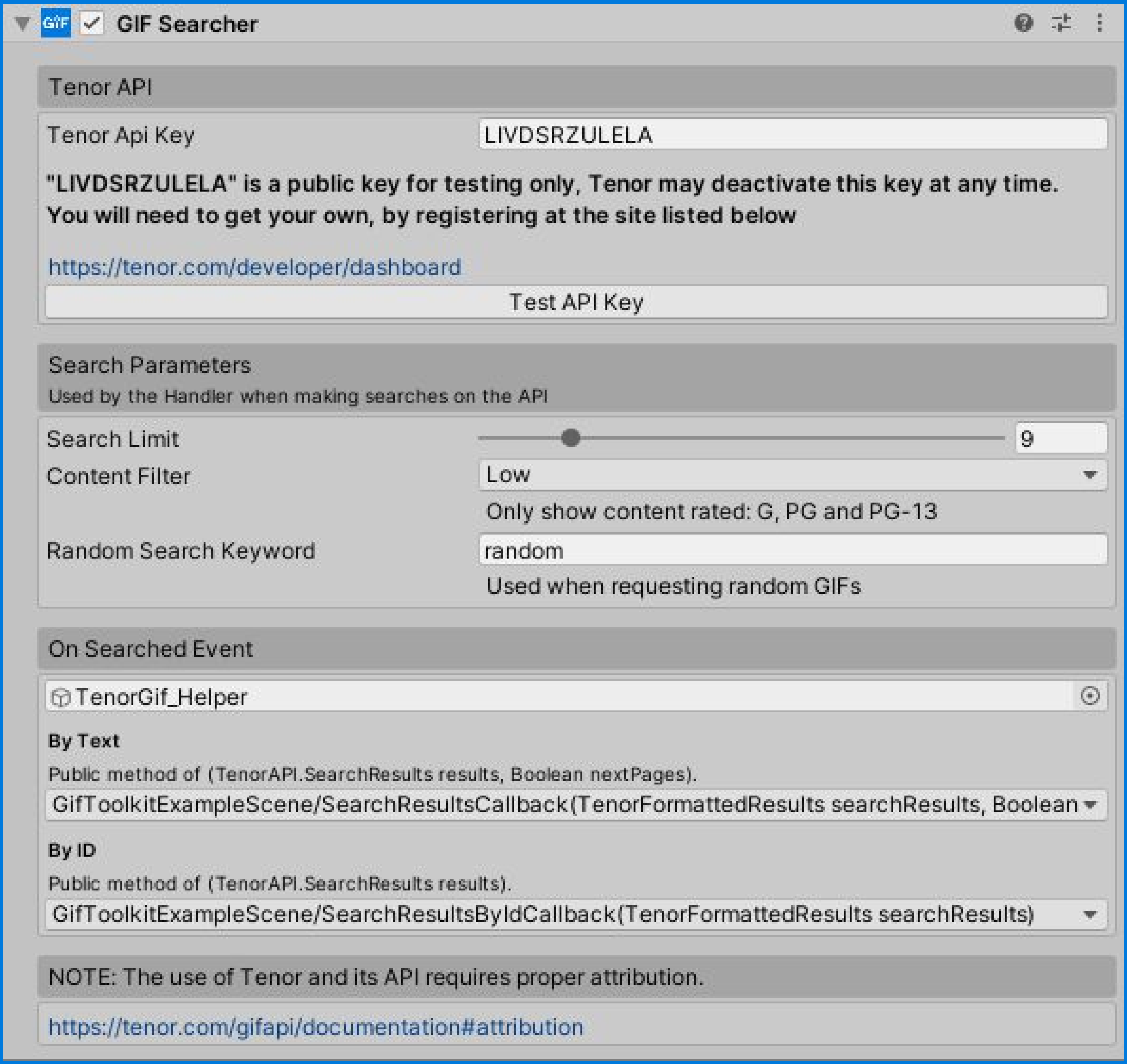
It is important that you attribute Tenor when using this plugin. As following the guidelines.
https://tenor.com/gifapi/documentation#attribution

A press kit has been provided for you in the Plugin. These images will be updated each time a new version of the plugin is released. However you can check yourself by visiting the link above.

You will need to acquire your own API Key when you start using the Plugin. The one included in the Plugin is a public facing key for testing purposes only, and could be revoked or changed at any time.
https://tenor.com/developer/dashboard

This Plugin comes in two main pieces, the CORE and the HANDLER. The Core is what drives the plugin, any data returned from Tenor is stored exactly as provided. The Handler uses Core to do the work, but wraps everything up with a useful Inspector component, and formatted results that make handling GIFs a lot easier. It is recommended you start off with Handler, then move onto Core if you need more control over the API. Please read the next section for more info and programming examples.

GifToolkit.Handler



GIF Searcher

Tenor API

Tenor Api Key: LIVDSRZULELA

"LIVDSRZULELA" is a public key for testing only, Tenor may deactivate this key at any time. You will need to get your own, by registering at the site listed below

https://tenor.com/developer/dashboard

Test API Key

Search Parameters

Used by the Handler when making searches on the API

Search Limit: 9

Content Filter: Low

Only show content rated: G, PG and PG-13

Random Search Keyword: random

Used when requesting random GIFs

On Searched Event

TenorGif_Helper

By Text

Public method of (TenorAPI.SearchResults results, Boolean nextPages).

GifToolkitExampleScene/SearchResultsCallback(TenorFormattedResults searchResults, Boolean

By ID

Public method of (TenorAPI.SearchResults results).

GifToolkitExampleScene/SearchResultsByIdCallback(TenorFormattedResults searchResults)

NOTE: The use of Tenor and its API requires proper attribution.

https://tenor.com/gifapi/documentation#attribution

Breakdown - GIF Searcher

GIF Searcher is the main component used with Handler. It has been added to the component menu under [GIF Toolkit/GIF Searcher]

Setting up in the inspector is as simple as adding your API key, setting some search parameters, and then pointing to some functions for returning results. More information on the results will be listed below.

Searching GIFS

Once the inspector component is set up correctly, you will be able to start making searches. If a pointer is not listed in the inspector, a warning message will be logged, and no search will execute.

Search GIFs

Here is a list of the following functions available for searching GIFs.
SearchGifs() and SearchGifsRandom() take a constructor of either String or UI Text.
So They can be called either from code, or from a UI element’s event.

RandomGifs() will do a search and return randomized results. The search will be based on the search term predefined in the Inspector. Great for loading a GIF search window with random GIFs before the User starts searching.

When SearchGifs(), SearchGifsRandom() or RandomGifs() is called, a reference to that search is stored. Then, when the function GetNextPage() is called, it will return additional results based on the last search.

A bool indicating whether the search is for a new page or not will be returned to the search callback function you define. That way you can handle either displaying a new set of GIFs, or adding to what is already displayed.

```
class Example
{
      GifToolkit.Handler.GifSearcher searcher;

      void DoSearch()
      {
            searcher.SearchGifs("Good Game");
            searcher.SearchGifsRandom("Good Game");
            searcher.GetNextPage();
            searcher.RandomGifs();
      }

      void SearchCallback(TenorFormattedResults searchResults, bool nextPages)
      {
            // Do something with search results...
      }
}
```

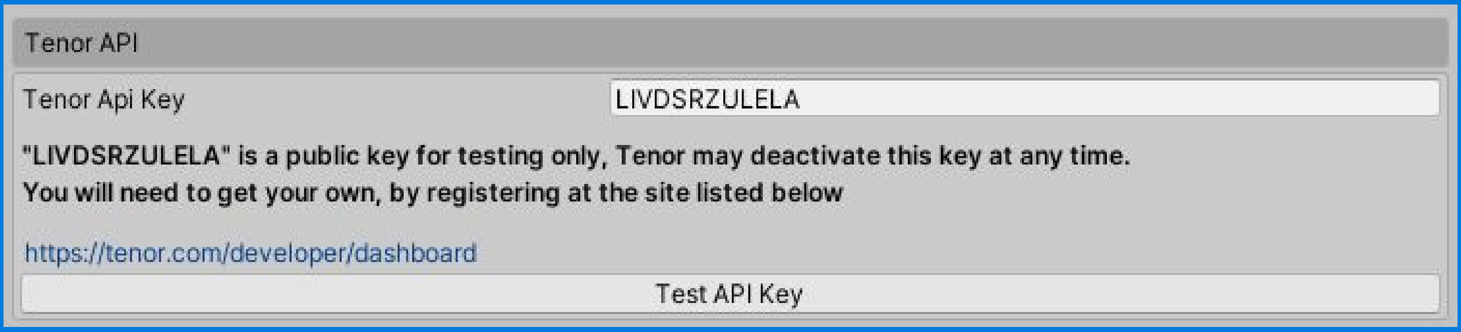
Get GIFs

You can get GIFs by ID with these two functions. The ID of a GIF is returned in the results when making a search. This is useful for sending minimal info of a GIF across the server, the remote client can receive the ID and then pull the GIF back down from Tenor. Rather than trying to send a byte[] of data then deserializing.

NOTE: These functions share a different callback when returning results. As you will see in the Inspector.
The callback does not require a bool indicating the next page, as it will always return the exact amount requested.
However in an instance where more than 50 GIF IDs are supplied, a warning message will be debugged and the requested results will be lowered to 50.

```
class Example
{
      GifToolkit.Handler.GifSearcher searcher;

      void DoSearch()
      {
            searcher.GetGifByID(7302854);
            searcher.GetGifsByIDs(new int[]{7302854, 15782889, 12532815, 8054142});
      }

      void SearchCallback(TenorFormattedResults searchResults)
      {
            // Do something with search results...
      }
}
```

Test API Key


Tenor API

Tenor Api Key: LIVDSRZULELA

"LIVDSRZULELA" is a public key for testing only, Tenor may deactivate this key at any time.
You will need to get your own, by registering at the site listed below

https://tenor.com/developer/dashboard

You can test a success of an API key from the inspector. A message in the console will be logged indicating its success. This is for development only, don’t call in game as it can hold up the UI.

Extras - The Helpers

```
using GifToolkit.Handler;
```

TenorFormattedResults

The formatted results are an adaptation of the SearchResults returned from searches in the Core API. In the constructor of TenorFormattedResults, you can pass the SearchResults and it will be automatically converted. You can also add additional results to the existing one with the AddPages() method. This is handy for keeping a reference if you have not finished with the previous result, and you're about to load more.

SelectGifByQuality

SelectGifByQuality is an extension method of the Media class. Media is the child of the TenorFormattedResults class and holds GIF information for all qualities. Useful for if you're giving the player decisions over what quality they want. An example..

```
class Example
{
      Media gifMedia;
      GifQualities quality;

      Gif GetGif(Media media)
      {
            if (quality = GifQualities.Nano)
                  return media.NanoSize;
            else if (quality = GifQualities.Tiny)
                  return media.TinySize;
            else
                  return media.NormalSize;
      }

      // VS the extension method
      Gif GetGif(Media media)
      {
            return media.SelectGifByQuality(quality);
      }
}
```

PrepareGifPlayer_NewRender and PrepareGifPlayer_ExisitngRender

As Unity does not support the .gif image format. GIFs are displayed through a video player. A video player targets a Render Texture, and a UI Raw Image uses the Render Texture.

Either of these functions will take an instance of a video player, then configure it and return with a new Render Texture to be used however you see fit. A good example of this is shown in the TenorGifExampleScene script.

GifToolkit.Core

The Core is a library written best to handle requests directly with the Tenor API. With core you can setup your own method of searching gifs. Handler will do almost everything you need, in an easier structured environment. For example, Handle only has two callback options. With Core you could write as many as you want. If you choose to use Core, it is still suggested that you use Handler's search result class to work with results.

SearchPramsAPI

This is a class that needs to be provided when making a search. Usage can be seen below in one of the search result examples.

NOTE: Search results are limited to a maximum of 50. When making a search, the SearchPramsAPI class will automatically clamp between 1 and 50 if you set it to anything lower or higher.

SearchPramsAPI references the following search requirements for the Tenor API

- Search Limit - Maximum Results
- Search Query - Name of GIFs your searching for
- Content Filter - Limits the result based on maturity. (P, PG, PG-13 and R)
- Next Page Pos - Indicates the position in which to start the search.
  When making a search, the results will return a "next" value. If you provide that in your next search as NextPagePos, then you will get the next lot of results.

TestApiKey

This is a static function that can be called at any time. It does not have any callback however so Unity will wait for the result before the UI is updated. Thus, only use this in development stages for testing your API key.

Searching

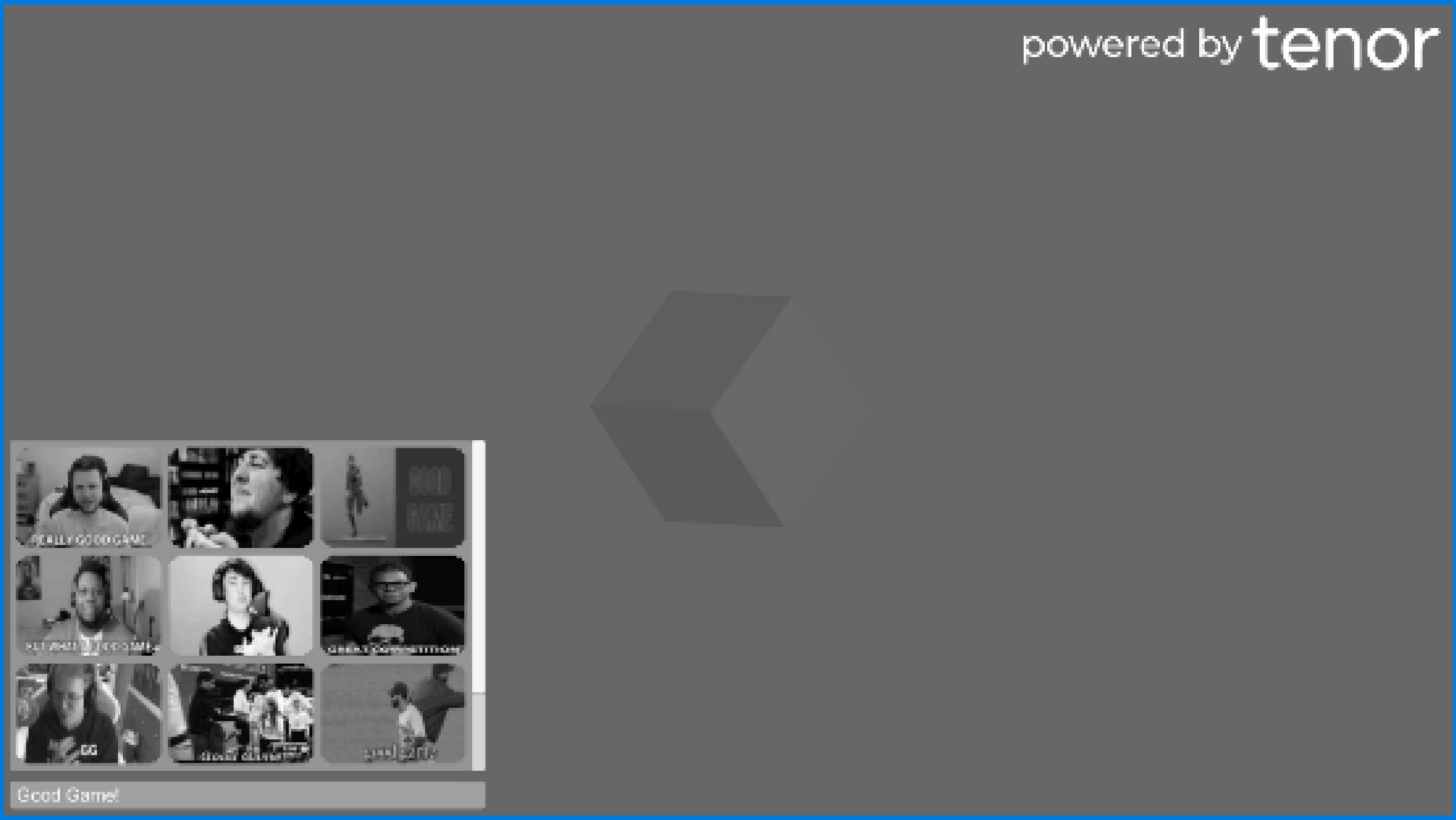
These are the following function available for searching:

- GetSearchResults()
- GetRandomSearchResults()
- GetGifByID()
- GetGifsByIDs()

All searches need to be called from StartCoroutine()
It is suggested that a lambda expression be used for making these search calls. Below will demonstrate the SearchPramsAPI, all search functions and the use of lambda vs normal.

```
class Example {
        GifToolkit.Core.TenoreServiceAPI api;
        GifToolkit.Core.SearchPramsAPI searchPrams = new GifToolkit.Core.SearchPramsAPI() {
                SearchQuery = "Good Game",
                ResultsLimit = 9,
                ContentFilter = ContentFilters.Medium
        };

        // Using lambda
        void Search(){
             StartCoroutine(api.GetSearchResults(searchPrams, (SearchResults searchResults) =>
             {
                 //Set this if you want to call again, and get the Next Page
                 searchPrams.NextPagePos = searchResults.next;
                 // Do Work Here
             }));
        }

        // Using lambda
        void SearchRandom(){
             StartCoroutine(api.GetRandomSearchResults(searchPrams, (SearchResults searchResults) =>
             {
                 //Set this if you want to call again, and get the Next Page
                 searchPrams.NextPagePos = searchResults.next;
                 // Do Work Here
             }));
        }

        // Not using lambda
        void GetByID(){
             int id = 8054142;
             StartCoroutine(api.GetGifByID(id, GetByIDCallback));
        }
        void GetByIDCallback(SearchResults.Result searchResults){
             // Do Work Here
        }

        // Not using lambda
        void GetByIDs(){
             int ids [] = new int[]{7302854, 15782889, 12532815, 8054142};
             StartCoroutine(api.GetGifsByIDs(ids, GetByIDSCallback));
        }
        void GetByIDSCallback(SearchResults.Result[] searchResults){
             // Do Work Here
        }
}
```

Example Scene



In the example scene, you will be shown a demonstration of how to search and post GIFs. The spinning cube in the scene is just proof that the API calls don't hold up Unity's UI.

When you run the scene, you will immediately be able send chat messages to an example in-game chat window. If you click the GIF button. A popup will open that pre-loads random GIFs. If you search a GIF, the results will be refreshed with what you have searched. Clicking the Load More button at the button of the results will bring additional searches.

It is highly encouraged to open up the GifToolkitExampleScene script and take a look. Especially the VramControll method. As this code is not directly related to the plugin, and may significantly change with an update, code won't be documented here. However the script itself is very well commented, and shows some good practices. It's easy to follow and will be enough to get you going. Once you're ready, the documentation for the Plugin itself will be here waiting for you.

Good luck, and have fun :)