This is a plugin based on the FastLZ library released under the MIT license (see above).

The scope of this library is to compress/decompress FastLZ archives and buffers on Android, iOS, tvOS*, Windows, OSX, Linux & webGL*.

The examples require some files that reside in the StreamingAssets folder. They are there just for testing purposes. In your final projects make sure you delete them so that you don't increase your build size.

The ios are compiled as universal and bitcode enabled. That means that they will support 32 and 64 bit builds.
(non-bitcode enabled iOS plugins are provided. If you are creating a non bitcode enabled project please use these provided plugins!)
OSX bundle is compiled now as 64-bit only, since Apple store requires it. An x86_64 + Silicon version is included as a zip. If you need silicon support, unzip and replace the .bundle.
The Windows and Linux libraries are compiled for x86 and x86_64 build modes.
The Android lib is compiled for armeabi-v7a, x86, x86_64 and arm64-v8a.

***webGL/tvOS support is for compressing/decompressing flz buffers only.**

The plugin will not work correct on files larger then 2GB.

FEATURES:

Fast FastLZ compression and decompression with a clean and simple interface. Very easy to use.

- compress a file into a FastLZ file format.

- decompress a FastLZ file.

- compress a buffer into the FastLZ format.

- decompress a FastLZ compressed buffer into a buffer.

(The plugin provides a solution to store the uncompressed size of a compressed buffer in its footer.)

- Linux, iOS, Android, MacOSX can treat buffers as files. That means if you have a file in www.bytes or in a native buffer you can perform operations directly on the buffer.
  For Android this is very useful since you can decompress from Streaming Assets without copying to Persistent data path.

!!! If you want to use only the FastLZ plugin, please delete all the other plugins in their respective folders or use the single packages from the _plugin_packages folder!!!.

INSTRUCTIONS:

If you want to run a small example, compile and run the testScene.
It will download a small tif file and it will perform all the functions the lib provides.

See the fLZ.cs file for more comments and error codes.

In your project include in the Plugins folder the plugins you want to use and the fLZ.cs file and call the appropriate functions as described below and shown in the demo scene.

THE FUNCTIONS:

```
int compressFile(string inFile, string outFile, int level, bool overwrite, ulong[] progress);
```

Compress a file to fLZ.

Full paths to the files should be provided.
Returns the size of the resulting archive in bytes.

level : level of compression (1 = faster/bigger, 2 = slower/smaller).
progress : provide a single item ulong array to get the progress of the compression in real time. (only when called from a thread)

```
int decompressFile(string inFile, string outFile, bool overwrite, ulong[] progress, object fileBuffer = null);
```

Decompress an fLZ file.

Full paths to the files should be provided.
Returns 1 on success.

progress : provide a single item ulong array to get the progress of the decompression in real time. (only when called from a thread/task)
fileBuffer : A buffer that holds an FLZ file. When assigned the function will decompress from this buffer and will ignore the filePath. (iOS, Android, MacOSX, Linux)
: It can be a byte[] buffer or a native IntPtr buffer (downloaded using the helper function: downloadFlzFileNative / see demo script)
: When an IntPtr is used as the input buffer, the size of it must be passed to the function as a string with the inFile parameter!

```
bool compressBuffer(byte[] inBuffer, ref byte[] outBuffer, int level, bool includeSize = true);
```

Compress a byte buffer in fLZ format.
Returns true on success.

inBuffer : the uncompressed buffer.
outBuffer : a referenced buffer that will be resized to fit the fLZ compressed data.
includeSize: include the uncompressed size of the buffer in the resulted compressed one because fLZ does not include this.
level: level of compression (1 = faster/bigger, 2 = slower/smaller).

```
byte[] compressBuffer(byte[] inBuffer, int level, bool includeSize = true);
```

Compress a byte buffer in fLZ format.
Returns a new buffer with the compressed data.

inBuffer : the uncompressed buffer.
outBuffer : a referenced buffer that will be resized to fit the fLZ compressed data.
includeSize: include the uncompressed size of the buffer in the resulted compressed one because fLZ does not include this.
level : level of compression (1 = faster/bigger, 2 = slower/smaller).

```
bool decompressBuffer(byte[] inBuffer, ref byte[] outBuffer, bool useFooter = true, int customLength = 0);
```

Decompress an fLZ compressed buffer to a referenced buffer.
Returns true on success.

inBuffer : the fLZ compressed buffer
outBuffer : a referenced buffer that will be resized to store the uncompressed data.
useFooter : if the input Buffer has the uncompressed size info.
customLength : provide the uncompressed size of the compressed buffer. Not needed if the useFooter is used!

```
byte[] decompressBuffer(byte[] inBuffer, bool useFooter = true, int customLength = 0);
```

Decompress an fLZ compressed buffer to a new buffer.
Returns a new buffer with the uncompressed data.

inBuffer : the fLZ compressed buffer
useFooter : if the input Buffer has the uncompressed size info.
customLength: provide the uncompressed size of the compressed buffer. Not needed if the useFooter is used!

```
int decompressBufferFixed(byte[] inBuffer, ref byte[] outBuffer, bool safe = true, bool useFooter =
true, int customLength = 0);
```

Decompress an flz compressed buffer to a referenced fixed size buffer.
Returns the uncompressedSize.

```
inBuffer    : the flz compressed buffer
outBuffer   : a referenced fixed size buffer where the data will get decompressed
useFooter   : if the input Buffer has the uncompressed size info.
customLength: provide the uncompressed size of the compressed buffer. Not needed if the useFooter is
              used!

This function is useful if you want to avoid memory allocations caused by new buffers or buffer
resizing.
```

```
[Android, iOS, Linux, MacOSX only]
int setFilePermissions(string filePath, string _user, string _group, string _other);
```

Sets permissions of a file in user, group, other.

Each string should contain any or all chars of "rwx".

Returns 0 on success.

Helper function

```
IEnumerator downloadFlzFileNative(string url, Action<bool> downloadDone, Action<IntPtr> pointer
= null, Action<int> fileSize = null);
```

A Coroutine to dowload a file to a native/unmaged memory buffer.
You can call it for an IntPtr.
See fLZtest.cs for usage example.

This function can only be called for one file at a time. Don't use it to call multiple files at once.

This is useful to avoid memory spikes when downloading large files and intend to decompress from memory.
With the old method, a copy of the downloaded file to memory would be produced by pinning the buffer to memory.
Now with this method, it is downloaded to memory and can be manipulated with no memory spikes.

In any case, if you don't need the created in-Memory file, you should use the fLZ.fLZreleaseBuffer function to free the memory!

Parameters:

url:	The url of the file you want to download to a native memory buffer.
downloadDone:	Informs a bool that the download of the file to memory is done.
pointer:	An IntPtr for a native memory buffer
fileSize:	The size of the downloaded file will be returned here.

SUPPORT:

For any questions, problems and suggestions please use this email address: elias_t@yahoo.com

forum: http://forum.unity3d.com/threads/7zip-lzma-and-zip-native-multiplatform-plugins.211273/