Large File Downloader

Large file downloader is a plugin for unity3d which can download any large file and save to disk. This plugin is using WebClient class to download the files.

Just import the plugin in your project and start using it by simply calling a function.

Android : You need to grant internet access permission to player settings, otherwise It will give NameResolutionFailure error.

NameResolutionFailure : This error will comes in case of no internet access or due to any security on the network.

If you have any queries then you can write me at devesh.pandey19@gmail.com

How to use:

1). Use LargeFileDownloader in your script where you want use it.

```
using LargeFileDownloader;
```

2). Create object of FileDownloader class.

```
FileDownloader downloader = new FileDownloader ();
```

3). Add http headers

There are few functions to add headers, so you can choose as per your need.

- public void AddHeader(string header)
- public void AddHeader(System.Collections.Specialized.NameValueCollection c)
- public void AddHeader(HttpResponseHeader header, string value)
- public void AddHeader(HttpRequestHeader header, string value)
- public void AddHeader(string name, string value)

For example : downloader.AddHeader("MY_HEADER_STRING");

4). Add callbacks.

```
FileDownloader.onComplete += OnDownloadComplete;
FileDownloader.onProgress += OnProgress;
```

This will call when downloading is completed for any file.

```
void OnDownloadComplete(DownloadEvent e)
{
	if(e.status == DownloadStatus.COMPLETED)
	{
		// File downloaded successfully
	}
	else if(e.status == DownloadStatus.CANCELLED)
	{
		// File cancelled
	}
	else if(e.status == DownloadStatus.FAILED)
	{
		// Downloading failed or there is any error
		if(e.error != null)
			Debug.Log("ERROR : "+e.error);
	}
}
```

This will call while file downloading.

```
void OnProgress(DownloadEvent e)
{
	if(e.status == DownloadStatus.PROGRESS)
	{
		// Progress of downloading
		// e.downloadedBytes : total bytes downloaded.
		// e.totalBytes : total bytes to be downloaded.
		// e.progress : progress in percentage (%).
		// e.fileURL : downloading file URL.
	}

}
```

5). Download a file.

i). downloader.Download (<<FILE_URL>>);
- It will download the file in Application.persistantDatapath.

ii). downloader.Download (<<FILE_URL>>, <<PATH_TO_SAVE>>);
FILE_URL : URL of file to download.
PATH_TO_SAVE : Path where downloaded file will be saved.

6). Add file in queue to download.

i). downloader.DownloadInQueue (<<FILE_URL>>);
- It will download the file in Application.persistantDatapath.

ii). downloader.DownloadInQueue (<<FILE_URL>>, <<PATH_TO_SAVE>>);
FILE_URL : URL of file to download.
PATH_TO_SAVE : Path where downloaded file will be saved.

7). Check if downloader is busy in downloading a file.

```
bool isBusy = downloader.IsBusy;
```

8). Check how many files are in queue to download.

```
int count = downloader.ItemsInQueue;
```

9). Check specific file is in downloading queue or not

```
bool isInQueue = downloader.IsInQueue(<<FILE_URL>>);
```

10). Cancel the current downloading file.

```
downloader.Cancel();
```

11). Cancel any file which is in downloading queue.

```
downloader.CancelInQueue(<<FILE_URL>>);
```

12). Cancel all

```
downloader.CancelAll();
```

13). Init the downloader again if you disposed it by calling Dispose function mentioned below.

```
downloader.Init();
```

Note : Add header after Init() if required.

14). Dispose the downloader instance. After calling this, if you want use downloader the you have to initialise it again by calling Init() function mentioned above.

```
downloader.Dispose();
```