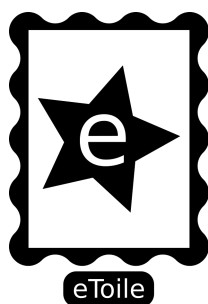




FILEMANAGEMENT CLASS

CROSS-PLATFORM FILE IO CONTROL WITHOUT PLUG-INS.



(eToile 2016) V: 1.3

Index

INTRODUCTION.....	4
CLASS DESCRIPTION.....	4
CLASS INTEGRATION.....	5
PLAYERPREFS REPLACEMENT.....	6
FILEMANAGEMENT PUBLIC INTERFACES.....	6
SAVERAWFILE().....	6
READRAWFILE().....	7
DELETEFILE().....	7
IMPORTAUDIO().....	7
FILEEXISTS().....	7
SAVEFILE().....	7
READFILE().....	8
SAVEARRAY().....	8
READARRAY().....	8
READLIST().....	9
IMPORTTEXTURE().....	9
IMPORTSPRITE().....	9
SAVEJPGTEXTURE().....	9
SAVEPNGTEXTURE().....	10
ADDLOGLINE().....	10
ADDRAWDATA().....	10
DIRECTORYEXISTS().....	10
CREATEDIRECTORY().....	11
DELETEDIRECTORY().....	11
EMPTYDIRECTORY().....	11
LISTFILES().....	11
LISTDIRECTORIES().....	11
READDIRECTORYCONTENT().....	12
COPYFILE().....	12
COPYDIRECTORY().....	12
MOVE().....	12
RENAME().....	13
GETPARENTDIRECTORY().....	13
COMBINE().....	13
NORMALIZEPATH().....	13
GETFILENAME().....	13

GETEXTENSION()	13
FILEMANAGEMENT PRIVATE INTERFACES	14
CUSTOMPARSER()	14
FILTERPATHNAMES()	14
SORTPATHNAMES()	14
ENCRYPT()	14
DECRYPT()	14
XORENCRYPTDECRYPT()	14
CHECKNAMEONINDEX()	14
GETNAMESONINDEX()	15
EXPERIMENTAL INTERFACES	15
AESENCRYPT()	15
AESDECRYPT()	15
OPENFOLDER()	15
OBJECTTOBYTEARRAY()	15
BYTEARRAYTOOBJECT()	15
LISTLOGICDRIVES()	16
AES ENCRYPTION	17
STREAMINGASSETS INDEXER	17
FILEBROWSER PREFAB	18
FILEBROWSER.CS	19
CONTENTITEM.CS	20
CUSTOMIZE THE BROWSER WINDOW	21
KNOWN ISSUES	23
CONTACT	23

INTRODUCTION

THANK YOU FOR PURCHASING `FileManagement`, THIS CLASS IS DESIGNED TO BE SIMPLE AND LIGHTWEIGHT, SO YOU WILL NOT NEED TO LEARN OR IMPLEMENT MORE THAN ITS USEFUL INTERFACES.

THIS PRODUCT IS JUST THAT: A CLASS, SO YOU CAN ADD IT TO YOUR PROJECT WITHOUT ANY RISKS.

YOU ALSO CAN ACCESS THE FULL SOURCE CODE.

SAVING AND READING IS AS FAST AS THE PLATFORM CAN ADMIT, BECAUSE INTERPRETATIONS AND PARSING ARE MAINTAINED AT MINIMUM.

THE EXAMPLE SCENE ALLOWS TESTING MOST OF THE `FileManagement` FUNCTIONS. THERE IS A VERY IMPORTANT FEATURE IN `FileManagement` THAT ALLOWS ACCESS `PersistentData` AND `StreamingAssets` AS A SINGLE DRIVE. THIS ALLOWS A COMPLETELY SAFE CROSS-PLATFORM DRIVE ACCESS.

CLASS DESCRIPTION

`FileManagement` IS A STATIC CLASS, THAT MEANS YOU DON'T HAVE TO CREATE/INSTANTIATE A `FileManagement` OBJECT, JUST WRITE `FileManagement` DOT (.) THE INTERFACE YOU NEED.

THE `FileManagement` CLASS USES COMPILER DIRECTIVES TO CHOSE THE RIGHT FUNCTIONS FOR EACH PLATFORM. SO, THERE ARE DIFFERENT VERSIONS FOR THE NEXT FUNCTIONS: `SaveRawFile()`, `ReadRawFile()` AND `DeleteFile()`.

THIS THREE FUNCTIONS ARE PLATFORM DEPENDENT, SO THEY ARE USED BY EVERY OTHER FUNCTION INTO THIS SAME CLASS.

THE THREE MAIN GROUPS ARE:

- `UNITY_WINRT`
- `UNITY_WEBGL`
- `EVERYTHING ELSE`

THERE ARE NO ANY SPECIAL CONSIDERATIONS WHEN EXPORTING TO OTHER PLATFORMS, NEITHER ANY SPECIAL CONSIDERATIONS WHEN UPLOADING TO DIGITAL MARKETS.

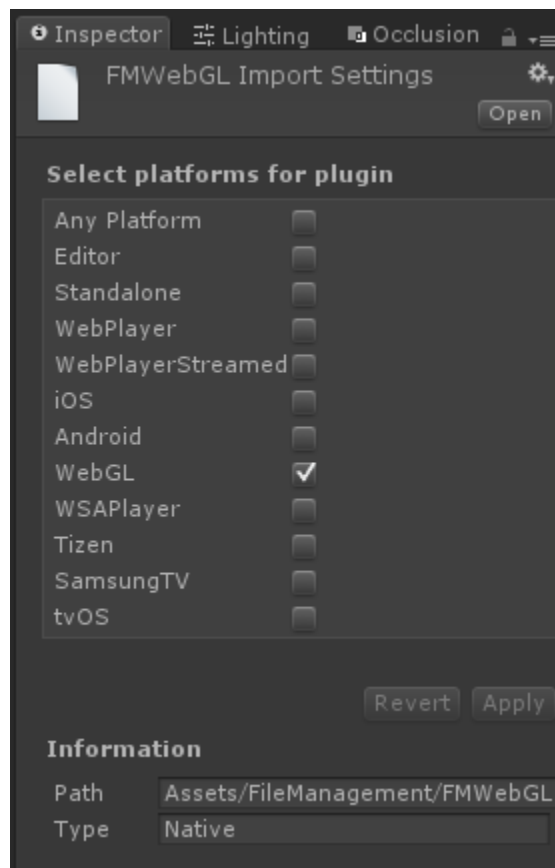
JUST SWITCH PLATFORM FROM "BUILD SETTINGS" DIALOG ON UNITY EDITOR.

CLASS INTEGRATION

TO INTEGRATE THIS CLASS TO YOUR PROJECT YOU MUST INCLUDE THE “FILEMANAGEMENT.CS” FILE ANYWHERE INTO YOUR UNITY PROJECT. AS THIS CLASS IS A STATIC CLASS YOU CAN ACCESS ITS CONTENT FROM ANY SCRIPT DIRECTLY.

THERE IS A PLUG-IN NEEDED FOR WebGL EXPORTS, THE “FMWebGL.JSLIB” FILE. THIS FILE CAN BE EDITED WITH ANY TEXT EDITOR, IT IS NOT A REAL PLUG-IN, IT IS SOME JAVASCRIPT EXTRA FUNCTIONALITY NEEDED TO SAVE CONTENT INTO BROWSER’S DATA BASE.

THE PLUG-IN IMPORTER SHOULD LOOK LIKE THIS:



NOTE THAT YOU DON’T NEED TO PUT THE PLUG-IN INTO A “PLUGINS” FOLDER.

THERE IS ANOTHER FILE NAMED STREAMINGASSETSINDEXER.CS THAT RUNS ONLY IN THE EDITOR GENERATING THE STREAMINGASSETS INDEX AUTOMATICALLY.

PLAYERPREFS REPLACEMENT

`FileManagement` IMPLEMENTS THE SAME `PlayerPrefs` FUNCTIONS, SO YOU ONLY NEED TO RENAME `PlayerPrefs` WITH `FileManagement`.

THIS IS THE LIST OF EQUIVALENT FUNCTIONS:

```
public static void DeleteAll()
public static void DeleteKey(string key)
public static float GetFloat(string key, float defaultValue = 0.0F)
public static int GetInt(string key, int defaultValue = 0)
public static string GetString(string key, string defaultValue = "")
public static bool HasKey(string key)
public static void Save()
public static void SetFloat(string key, float value)
public static void SetInt(string key, int value)
public static void SetString(string key, string value)
```

THE `Save()` FUNCTION HAS NO EFFECT IN `FileManagement` DUE TO THERE IS NO PARSING OF VALUES INTO A SINGLE FILE. THE INTERFACE EXISTS JUST TO AVOID COMPILATION ISSUES WHEN PORTING AN EXISTING APPLICATION.

THIS IS THE LIST OF NON STANDARD FUNCTIONS:

```
public static bool GetBool(string key, bool defaultValue = false)
public static double GetDouble(string key, double defaultValue = 0)
public static void SetBool(string key, bool value)
public static void SetDouble(string key, double value)
```

THESE FUNCTIONS WORKS IN THE SAME WAY THAT `PlayerPrefs` DOES BUT ADDS SOME MISSING FUNCTIONALITY.

FILEMANAGEMENT PUBLIC INTERFACES

THIS IS THE COMPLETE DEFINITION OF `FileManagement` PUBLIC INTERFACES.

SAVERAWFILE()

```
public static void SaveRawFile(string name, byte[] content, bool enc = false,
bool fullPath = false)
```

USE A KEY, ID OR FILE NAME TO IDENTIFY YOUR FILE.

THE `enc` ARGUMENT ENABLES OR DISABLES THE ENCRYPTION FUNCTIONALITY.

THE `fullPath` ARGUMENT ALLOWS TO TREAT THE PROVIDED `name` AS A PATH.

THE NAME CAN INCLUDE A RELATIVE OR ABSOLUTE DESTINATION PATH. IF THAT PATH DOESN'T EXISTS IT WILL BE CREATED AUTOMATICALLY.

NOTE: THIS FUNCTION IS USED BY EVERY OTHER FUNCTION THAT WRITES TO DISK.

THIS EXAMPLE SAVES A BINARY FILE:

```
byte[] byteArray = {10, 20, 30, 40, 50};
FileManagement.SaveRawFile("data.bin", byteArray);
```

READRAWFILE()

```
public static byte[] ReadRawFile(string name, bool enc = false, bool checkSA = true, bool fullPath = false)
```

USE A KEY, ID OR FILE NAME TO IDENTIFY YOUR FILE. IT RETURNS EMPTY CONTENT IF THE FILE DO NOT EXISTS.

THE enc ARGUMENT ENABLES OR DISABLES THE ENCRYPTION FUNCTIONALITY.

THE checkSA ARGUMENT ALLOWS [FileManagement](#) TO SEARCH THE

“STREAMINGASSETS” FOLDER IF THE REQUESTED FILE IS NOT FOUND.

THE fullPath ARGUMENT ALLOWS TO TREAT THE PROVIDED name AS A PATH.

THIS EXAMPLE READS AN ENCRYPTED TEXT FILE

(THE CONTENT VALUE IS DECRYPTED):

```
byte[] content = FileManagement.ReadRawFile ("data.txt", true);
```

DELETEFILE()

```
public static void DeleteFile(string name, bool fullPath = false)
```

DELETES THE FILE. IT FAILS IF THE ACCESS IS DENIED USING fullPath.

THE fullPath ARGUMENT ALLOWS TO TREAT THE PROVIDED name AS A PATH.

THIS EXAMPLE DELETES THE FILE:

```
FileManagement.DeleteFile("data.txt");
```

IMPORTAUDIO()

```
public static AudioClip ImportAudio(string file, bool enc = false, bool checkSA = true, bool fullPath = false)
```

IMPORTS AN AUDIOCLIP FROM FILE. THE WAV FORMAT IS THE ONLY WIDELY SUPPORTED BY ALL PLATFORMS, FOR OTHER FORMATS CHECK THE UNITY DOCUMENTATION (OR TRY THE TEST APPLICATION IN YOUR DESIRED PLATFORM).

THIS FUNCTION IMPLEMENTS A WAV FILE PARSER TO ALLOW WebGL SUPPORT. ONLY 16BIT WAV FILES ARE SUPPORTED (MONO OR STEREO, 22K, 44K, ETC.).

FILEEXISTS()

```
public static bool FileExists(string name, bool checkSA = false, bool fullPath = false)
```

CHECKS IF THE FILE EXISTS.

THE checkSA ARGUMENT ALLOWS [FileManagement](#) TO SEARCH THE STREAMINGASSETS FOLDER IF THE REQUESTED FILE IS NOT FOUND.

THE fullPath ARGUMENT ALLOWS TO TREAT THE PROVIDED name AS A PATH.

THIS EXAMPLE CHECKS IF A FILE EXISTS:

```
if (FileManagement.FileExists("data.txt"))  
    Debug.Log("[FileManagement] This file exists.");
```

SAVEFILE()

```
public static void SaveFile<T>(string name, T content, bool enc = false, bool fullPath = false)
```

SAVES ANY C# TYPE OF VARIABLE, AND THE FOLLOWING [UnityEngine](#) TYPES:

Vector2, Vector3, Vector4, Quaternion, Rect, Color & Color32.

THE enc ARGUMENT ENABLES OR DISABLES THE ENCRYPTION FUNCTIONALITY.

THE fullPath ARGUMENT ALLOWS TO TREAT THE PROVIDED name AS A PATH.

THIS EXAMPLE SAVES TWO TYPES OF DATA:

```
FileManagement.SaveFile("IntData", 12);  
FileManagement.SaveFile("StringData", "Example data", true); // Encrypt
```

READFILE()

```
public static T ReadFile<T>(string name, bool enc = false, bool checkSA = false,  
bool fullPath = false)
```

READS ANY C# TYPE OF VARIABLE, AND READS THE FOLLOWING UnityEngine

TYPES: Vector2, Vector3, Vector4, Quaternion, Rect, Color & Color32.

THE enc ARGUMENT ENABLES OR DISABLES THE ENCRYPTION FUNCTIONALITY.

THE checkSA ARGUMENT ALLOWS FileManagement TO SEARCH THE STREAMINGASSETS FOLDER IF THE REQUESTED FILE IS NOT FOUND.

THE fullPath ARGUMENT ALLOWS TO TREAT THE PROVIDED name AS A PATH.

THIS EXAMPLE READS TWO DIFFERENT TYPES OF FILE:

```
int a = FileManagement.ReadFile<int>("IntData");  
string a = FileManagement.ReadFile<string>("StringData", true); // Decrypt
```

SAVEARRAY()

```
public static void SaveArray<T>(string name, T[] content, char separator = (char)0x09,  
bool enc = false, bool fullPath = false)
```

```
public static void SaveArray<T>(string name, System.Collections.Generic.List<T> content,  
char separator = (char)0x09, bool enc = false, bool fullPath = false)
```

SAVES ANY ONE DIMENSION Array OR List OF ANY ReadFile SUPPORTED TYPE.

YOU CAN SPECIFY A CUSTOM SEPARATOR AS A char ARGUMENT, DEFAULT SEPARATOR IS THE HORIZONTAL TABULATOR.

THE enc ARGUMENT ENABLES OR DISABLES THE ENCRYPTION FUNCTIONALITY.

THE fullPath ARGUMENT ALLOWS TO TREAT THE PROVIDED name AS A PATH.

THIS EXAMPLE SAVES AN ARRAY OF STRINGS SEPARATED BY A SEMICOLON:

```
string[] myArray = {"one", "two", "three"};  
FileManagement.SaveArray("MyArray.csv", myArray, ';');
```

READARRAY()

```
public static T[] ReadArray<T>(string name, char separator = (char)0x09, bool enc = false,  
bool checkSA = true, bool fullPath = false)
```

READS ANY ONE DIMENSION Array OF ANY ReadFile SUPPORTED TYPE.

YOU CAN SPECIFY A CUSTOM SEPARATOR AS A char ARGUMENT, DEFAULT SEPARATOR IS THE HORIZONTAL TABULATOR.

THE enc ARGUMENT ENABLES OR DISABLES THE ENCRYPTION FUNCTIONALITY.

THE checkSA ARGUMENT ALLOWS FileManagement TO SEARCH THE STREAMINGASSETS FOLDER IF THE REQUESTED FILE IS NOT FOUND.

THE fullPath ARGUMENT ALLOWS TO TREAT THE PROVIDED name AS A PATH.

THIS EXAMPLE READS AN ARRAY OF STRINGS SEPARATED BY A SEMICOLON:

```
string[] myArray = FileManagement.ReadArray<string>("MyArray.csv", ';');
```


READLIST()

```
public static System.Collections.Generic.List<T> ReadList<T>(string name,  
char separator = (char)0x09, bool enc = false, bool checkSA = true, bool fullPath = false)
```

READS ANY ONE DIMENSION `List` OF ANY `ReadFile` SUPPORTED TYPE.

YOU CAN SPECIFY A CUSTOM SEPARATOR AS A `char` ARGUMENT, DEFAULT SEPARATOR IS THE HORIZONTAL TABULATOR.

THE `enc` ARGUMENT ENABLES OR DISABLES THE ENCRYPTION FUNCTIONALITY.

THE `checkSA` ARGUMENT ALLOWS `FileManagement` TO SEARCH THE `STREAMINGASSETS` FOLDER IF THE REQUESTED FILE IS NOT FOUND.

THE `fullPath` ARGUMENT ALLOWS TO TREAT THE PROVIDED `name` AS A PATH.

THIS EXAMPLE READS AN ARRAY OF STRINGS SEPARATED BY A SEMICOLON:

```
string[] myArray = FileManagement.ReadArray<string>("MyArray.csv", ';');
```

IMPORTTEXTURE()

```
public static Texture2D ImportTexture(string file, bool enc = false, bool checkSA = true,  
bool fullPath = false)
```

IMPORTS A JPG OR PNG IMAGE FILE FROM DISK INTO A UNITY TEXTURE.

THE `enc` ARGUMENT ENABLES OR DISABLES THE ENCRYPTION FUNCTIONALITY, IT IS DISABLED BY DEFAULT.

THE `enc` ARGUMENT ENABLES OR DISABLES THE ENCRYPTION FUNCTIONALITY.

THE `checkSA` ARGUMENT ALLOWS `FileManagement` TO SEARCH THE `STREAMINGASSETS` FOLDER IF THE REQUESTED FILE IS NOT FOUND.

THE `fullPath` ARGUMENT ALLOWS TO TREAT THE PROVIDED `name` AS A PATH.

THIS EXAMPLE IMPORTS AN IMAGE FILE INTO A TEXTURE FROM THE `STREAMINGASSETS` FOLDER:

```
Texture2D texture = FileManagement.ImportTexture("image.jpg");
```

IMPORTSPRITE()

```
public static Sprite ImportSprite(string file, bool enc = false, bool checkSA = true,  
bool fullPath = false)
```

IMPORTS A JPG OR PNG IMAGE FILE FROM DISK INTO A UNITY SPRITE.

THE `enc` ARGUMENT ENABLES OR DISABLES THE ENCRYPTION FUNCTIONALITY.

THE `checkSA` ARGUMENT ALLOWS `FileManagement` TO SEARCH THE `STREAMINGASSETS` FOLDER IF THE REQUESTED FILE IS NOT FOUND.

THE `fullPath` ARGUMENT ALLOWS TO TREAT THE PROVIDED `name` AS A PATH.

THIS EXAMPLE IMPORTS AN IMAGE FILE INTO A SPRITE FROM THE `STREAMINGASSETS` FOLDER:

```
Sprite sprite = FileManagement.ImportSprite("image.jpg");
```

SAVEJPGTEXTURE()

```
public static void SaveJpgTexture(string name, Texture texture, int quality = 75,  
bool enc = false, bool fullPath = false)
```

SAVES A `Texture` OR `Texture2D` TO A FILE ENCODED IN JPG FORMAT. THE QUALITY PARAMETER DETERMINES THE FILE COMPRESSION, THE DEFAULT FILE COMPRESSION IS 75.

THE enc ARGUMENT ENABLES OR DISABLES THE ENCRYPTION FUNCTIONALITY.
THE fullPath ARGUMENT ALLOWS TO TREAT THE PROVIDED name AS A PATH.

THIS EXAMPLES SAVES TEXTURES INTO FILES:

```
FileManagement.SaveJpgTexture("texture.jpg", renderer.material.mainTexture);  
FileManagement.SaveJpgTexture("texture.jpg", gameObject.GetComponent<Sprite>().texture);
```

SAVEPNGTEXTURE()

```
public static void SavePngTexture(string name, Texture texture, bool enc = false,  
bool fullPath = false)
```

SAVES A [Texture](#) OR [Texture2D](#) TO A FILE ENCODED IN PNG FORMAT.

THE enc ARGUMENT ENABLES OR DISABLES THE ENCRYPTION FUNCTIONALITY.

THE fullPath ARGUMENT ALLOWS TO TREAT THE PROVIDED name AS A PATH.

THIS EXAMPLES SAVES TEXTURES INTO FILES:

```
FileManagement.SavePngTexture("texture.png", renderer.material.mainTexture);  
FileManagement.SavePngTexture("texture.png", sprite.texture);
```

ADDLOGLINE()

```
public static void AddLogLine(string name, string content, bool deleteDate = false,  
bool enc = false, bool fullPath = false)
```

THIS ADDS A SINGLE LINE OF TEXT TO AN EXISTING FILE SAVING ALSO
DATE AND TIME. USED FOR LOG FILES AND ERROR TRACKING.

THE deleteDate VARIABLE DISABLES THE AUTOMATIC PRINTING OF FORMATTED
DATE AND TIME FOR EACH NEW LINE.

THE enc ARGUMENT ENABLES OR DISABLES THE ENCRYPTION FUNCTIONALITY.

THE fullPath ARGUMENT ALLOWS TO TREAT THE PROVIDED name AS A PATH.

THIS EXAMPLES SAVES TEXTURES INTO FILES:

```
FileManagement.SavePngTexture(renderer.material.mainTexture, "texture.png");  
FileManagement.SavePngTexture(sprite.texture, "texture.png");
```

ADDRAWDATA()

```
public static void AddRawData(string name, byte[] content, bool enc = false,  
bool fullPath = false)
```

THIS ADDS A CHUNK OF [byte](#) DATA TO AN EXISTING FILE. IF THE REQUESTED
FILE DOESN'T EXISTS, A NEW FILE WILL BE CREATED.

THE enc ARGUMENT ENABLES OR DISABLES THE ENCRYPTION FUNCTIONALITY.

THE fullPath ARGUMENT ALLOWS TO TREAT THE PROVIDED name AS A PATH.

THIS EXAMPLES SAVES TEXTURES INTO FILES:

```
FileManagement.SavePngTexture(renderer.material.mainTexture, "texture.png");  
FileManagement.SavePngTexture(sprite.texture, "texture.png");
```

DIRECTORYEXISTS()

```
public static bool DirectoryExists(string folder = "", bool checkSA = true,  
bool fullPath = false)
```

CHECKS THE EXISTENCE OF A DIRECTORY.

THE checkSA ARGUMENT ALLOWS [FileManagement](#) TO SEARCH THE STREAMINGASSETS
FOLDER IF THE REQUESTED PATH IS NOT FOUND.

THE fullPath ARGUMENT ALLOWS TO TREAT THE PROVIDED name AS A PATH.

THIS EXAMPLE CHECKS IF A DIRECTORY EXISTS:

```
if (FileManagement.DirectoryExists ("Test1"))  
    Debug.Log("[FileManagement] This folder exists.");
```

CREATEDIRECTORY()

```
public static void CreateDirectory(string name, bool fullPath = false)
```

CREATES A NEW DIRECTORY. FAILS IF THE ACCESS IS DENIED USING fullPath.

THE fullPath ARGUMENT ALLOWS TO TREAT THE PROVIDED name AS A PATH.

THIS EXAMPLE CREATES A NEW DIRECTORY:

```
FileManagement.CreateDirectory("Test1");
```

DELETEDIRECTORY()

```
public static void DeleteDirectory(string name, bool fullPath = false)
```

DELETES A DIRECTORY AND ITS CONTENT INCLUDING SUB-DIRECTORIES. FAILS IF THE ACCESS IS DENIED USING fullPath.

THE fullPath ARGUMENT ALLOWS TO TREAT THE PROVIDED name AS A PATH.

THIS EXAMPLE DELETES AN EXISTING DIRECTORY:

```
FileManagement.DeleteDirectory("Test1");
```

EMPTYDIRECTORY()

```
public static void EmptyDirectory(string name = "", bool filesOnly = true,  
bool fullPath = false)
```

DELETES THE DIRECTORY CONTENT. BY DEFAULT DELETES ONLY FILES. FAILS IF THE ACCESS IS DENIED USING fullPath.

THE fullPath ARGUMENT ALLOWS TO TREAT THE PROVIDED name AS A PATH.

THIS EXAMPLE DELETES THE WHOLE CONTENT OF A FOLDER:

```
FileManagement.EmptyDirectory("Test1", false);
```

LISTFILES()

```
public static string[] ListFiles(string folder = "", bool checkSA = true,  
bool fullPath = false)
```

RETURNS ALL OF THE FILE NAMES CONTAINED IN THE REQUESTED FOLDER. FAILS IF THE ACCESS IS DENIED USING fullPath.

THE checkSA ARGUMENT ALLOWS FileManagement TO SEARCH THE STREAMINGASSETS FOLDER IF THE REQUESTED PATH IS NOT FOUND.

THE fullPath ARGUMENT ALLOWS TO TREAT THE PROVIDED name AS A PATH.

THIS EXAMPLE REQUESTS THE FOLDER CONTENT:

```
string[] fileNames = FileManagement.ListFiles("Test1");
```

LISTDIRECTORIES()

```
public static string[] ListDirectories(string folder = "", bool checkSA = true,  
bool fullPath = false)
```

RETURNS ALL OF THE FOLDER NAMES CONTAINED IN THE REQUESTED FOLDER. FAILS IF THE ACCESS IS DENIED USING fullPath.

THE checkSA ARGUMENT ALLOWS `FileManagement` TO SEARCH THE `STREAMINGASSETS` FOLDER IF THE REQUESTED PATH IS NOT FOUND.

THE `fullPath` ARGUMENT ALLOWS TO TREAT THE PROVIDED name AS A PATH.

THIS EXAMPLE REQUESTS THE FOLDER CONTENT:

```
string[] folderNames = FileManagement.ListDirectories("Test1");
```

READDIRECTORYCONTENT()

```
public static System.Collections.Generic.List<byte[]> ReadDirectoryContent  
(string folder = "", bool enc = false, bool checkSA = true, bool fullPath = false)
```

RETURNS ALL OF THE FILES CONTAINED IN THE REQUESTED FOLDER AS A `List` OF `byte` ARRAYS. THE FILES ARE ADDED TO THE `List` IN THE SAME ORDER THAT ARE LISTED WITH THE `ListFiles` FUNCTION.

FAILS IF THE ACCESS IS DENIED USING `fullPath`.

THE `enc` ARGUMENT ENABLES OR DISABLES THE ENCRYPTION FUNCTIONALITY.

THE checkSA ARGUMENT ALLOWS `FileManagement` TO SEARCH THE `STREAMINGASSETS` FOLDER IF THE REQUESTED PATH IS NOT FOUND.

THE `fullPath` ARGUMENT ALLOWS TO TREAT THE PROVIDED name AS A PATH.

THIS EXAMPLE REQUESTS THE FOLDER CONTENT:

```
List<byte[]> files = FileManagement.ReadDirectoryContent("Test1");  
string[] fileNames = FileManagement.ListFiles("Test1");
```

COPYFILE()

```
public static void CopyFile(string source, string dest, bool checkSA = true,  
bool fullPathSource = false, bool fullPathDest = false)
```

COPIES A FILE FROM `source` TO `dest`. THE DESTINATION PATH CAN INCLUDE A NEW NAME FOR THE COPIED FILE.

THE `checkSA` PARAMETER ONLY AFFECTS THE `source` PATH.

THIS EXAMPLE COPIES A FILE:

```
FileManagement.CopyFile ("data.txt", "NewFolder/dataCopy.txt");
```

COPYDIRECTORY()

```
public static void CopyDirectory(string source, string dest, bool checkSA = true,  
bool fullPathSource = false, bool fullPathDest = false)
```

COPIES A FILE FROM `source` TO `dest`. THE DESTINATION PATH CAN INCLUDE A NEW NAME FOR THE COPIED FILE.

IF THE `dest` PATH DOESN'T EXISTS, IT WILL BE CREATED AUTOMATICALLY.

THIS EXAMPLE COPIES A FILE:

```
FileManagement.CopyFile ("data.txt", "NewFolder/dataCopy.txt");
```

MOVE()

```
public static void Move(string source, string dest, bool fullPathSource = false,  
bool fullPathDest = false)
```

MOVES A FILE OR FOLDER FROM `source` TO `dest`. THE FOLDERS ARE MOVED WITH ALL ITS CONTENT.

THIS EXAMPLE MOVES A FOLDER AND ALL ITS CONTENT:

```
FileManagement.Move ("NewFolder/Test2", "NewFolder2/Test2");
```

RENAME()

```
public static void Rename(string source, string dest, bool fullPathSource = false,
bool fullPathDest = false)
```

THIS FUNCTION IS THE SAME AS Move(), YOU CAN USE IT TO RENAME FILES AND FOLDERS.

THIS EXAMPLE MOVES A FOLDER AND ALL ITS CONTENT:

```
FileManagement.Move ("NewFolder/Test2", "NewFolder2/Test2");
```

GETPARENTDIRECTORY()

```
public static string GetParentDirectory(string path = "")
```

RETURNS A VALID PATH BUT REMOVING THE FILE OR FOLDER.

THIS EXAMPLE REQUEST A PARENT FOLDER:

```
string parentPath = FileManagement.GetParentDirectory ("Test1/Test2");
```

NOW PARENTPATH'S VALUE IS "TEST1".

COMBINE()

```
public static string Combine(string path1, string path2)
```

RETURNS A VALID PATH COMBINING CORRECTLY BOTH PATHS.

THIS EXAMPLE COMBINES A PATH AND A FILE:

```
string path = FileManagement.Combine("Test1\\Test2", "icon2.png");
```

NOW PATH'S VALUE IS "TEST1/TEST2/ICON2.PNG".

NORMALIZEPATH()

```
public static string NormalizePath(string path)
```

RETURNS A VALID PATH, CORRECTING POSSIBLE ERRORS IN THE STRING.

THE FileManagement CLASS USES SLASHES, DELETES THE ENDING SLASH IN A PATH AND REPLACES DOUBLE SLASHES.

THIS EXAMPLE RETURNS A VALID DIRECTORY PATH:

```
string path = FileManagement.NormalizePath ("Test1\\Test2\\");
```

NOW PATH'S VALUE IS "TEST1/TEST2".

GETFILENAME()

```
public static string GetFileName(string path)
```

RETURNS THE LAST NAME OF THE PATH. IT CAN BE A FILE OR A FOLDER.

THIS EXAMPLE RETURNS THE FILE NAME ONLY:

```
string name = FileManagement.GetFileName ("Test1\\Test2\\data5.txt");
```

NOW NAME'S VALUE IS "DATA5.TXT".

GETEXTENSION()

```
public static string GetExtension(string path)
```

RETURNS THE FILE EXTENSION OF THE PATH (IF ANY). IF THERE IS NO EXTENSION IT WILL RETURN AN EMPTY STRING.

THIS EXAMPLE RETURNS THE FILE EXTENSION:

```
string extension = FileManagement.GetExtension ("Test1\\Test2\\icon2.png");
```

NOW PATH'S VALUE IS ".PNG".

FILEMANAGEMENT PRIVATE INTERFACES

THIS IS THE COMPLETE DEFINITION OF `FileManagement` PRIVATE INTERFACES.

CUSTOMPARSER()

```
private static T CustomParser<T>(string content)
```

THIS FUNCTION IS RESPONSIBLE OF CONVERTING THE TEXT SAVED DATA INTO THE CORRECT DATA AGAIN. IT SUPPORTS: EVERY C# TYPE, `Vector2`, `Vector3`, `Vector4`, `Quaternion`, `Rect`, `Color` & `Color32`.

THIS PARSER IS USED INTO `ReadFile` AND `ReadList`.

FILTERPATHNAMES()

```
private static string[] FilterPathNames(string[] names)
```

THIS FUNCTION DOES THE INVERSE AS `GetParentDirectory` BECAUSE RETURNS THE FILE OR FOLDER NAME REMOVING THE PARENT.

ITS USE IS INTERNAL, SO IT'S PRIVATE.

SORTPATHNAMES()

```
private static string[] FilterPathNames(string[] names)
```

THIS FUNCTION SORTS THE NAMES BY ALPHABET.

ITS USE IS INTERNAL, SO IT'S PRIVATE.

ENCRYPT()

```
private static byte[] Encrypt(byte[] data, byte[] key)
```

THIS IS THE FUNCTION THAT MAKES THE ENCRYPTION OF A FILE CONTENT. IT ONLY WORKS WITH BYTE ARRAYS. THIS FUNCTION IS LOCAL AND YOU DON'T NEED TO USE IT DIRECTLY. THIS FUNCTION CAN BE UPDATED WITH YOUR FAVORITE ENCRYPTION ALGORITHM.

DECRYPT()

```
private static byte[] Decrypt(byte[] data, byte[] key)
```

THIS IS THE FUNCTION THAT MAKES THE DECRYPTION OF A FILE CONTENT. IT ONLY WORKS WITH BYTE ARRAYS. THIS FUNCTION IS LOCAL AND YOU DON'T NEED TO USE IT DIRECTLY. THIS FUNCTION CAN BE UPDATED WITH YOUR FAVORITE ENCRYPTION ALGORITHM.

XORENCRYPTDECRYPT()

```
private static byte[] XorEncryptDecrypt(byte[] data, byte[] key)
```

PERFORMS XOR ENCRYPTION AND DECRYPTION.

CHECKNAMEONINDEX()

```
private static bool CheckNameOnIndex(string name, string type)
```

CHECKS THE NAME INTO DE `STREAMINGASSETS` FOLDER AUTOMATIC INDEX. EMULATES THE `FileExists` AND `DirectoryExists` FUNCTIONS FOR ANDROID AND

WEBGL BUILDS. THERE ARE TWO SLIGHTLY DIFFERENT VERSIONS OF THIS FUNCTION FOR EACH PLATFORM (ANDROID AND WEBGL).

GETNAMESONINDEX()

`private static string[] GetNamesOnIndex(string name, string type)`

CHECKS THE NAME INTO DE STREAMINGASSETS FOLDER AUTOMATIC INDEX AND RETURNS THE LIST OF THE ASSOCIATED NAMES. EMULATES THE ListFiles AND ListDirectories FUNCTIONS FOR ANDROID AND WEBGL BUILDS. THERE ARE TWO SLIGHTLY DIFFERENT VERSIONS OF THIS FUNCTION FOR EACH PLATFORM (ANDROID AND WEBGL).

EXPERIMENTAL INTERFACES

THE EXPERIMENTAL INTERFACES ARE NOT 100% SUPPORTED BY ALL PLATFORMS. `FileManagement` WILL PRINT IN CONSOLE IF THE SELECTED PLATFORM HASN'T THE REQUESTED FUNCTIONALITY.

AESENCRYPT()

`private static byte[] AesEncrypt(byte[] data, byte[] key)`

PERFORMS AES ENCRYPTION. NOT SUPPORTED ON WINDOWS STORE.

AESDECRYPT()

`private static byte[] AesDecrypt(byte[] data, byte[] key)`

PERFORMS AES DECRYPTION. NOT SUPPORTED ON WINDOWS STORE.

OPENFOLDER()

`public static void OpenFolder(string path = "", bool fullPath = false)`

OPENS THE REQUESTED FOLDER IN THE DEFAULT FILE SYSTEM. THIS FUNCTIONS WORKS IN STANDALONE BUILDS ONLY (WINDOWS, LINUX, OSX).

OBJECTTOBYTEARRAY()

`private static byte[] ObjectToByteArray(object obj)`

SERIALIZES AN OBJECT TO A BYTE ARRAY ALLOWING TO BE SAVED TO DISK WITH `SaveRawFile`.

PLEASE NOTE THAT NOT EVERY VARIABLE/CLASS TYPE CAN BE SERIALIZED DIRECTLY. THAT'S THE REASON THIS IS NOT THE DEFAULT SAVE METHOD.

THIS FUNCTION IS NOT SUPPORTED IN WINDOWS STORE BUILDS.

BYTEARRAYTOOBJECT()

`public static object ByteArrayToObject(byte[] arrBytes)`

DESERIALIZES A BYTE ARRAY INTO AN OBJECT ALLOWING TO BE LOADED

FROM DISK WITH ReadRawFile.

PLEASE NOTE THAT NOT EVERY VARIABLE/CLASS TYPE CAN BE SERIALIZED DIRECTLY. THAT'S THE REASON THIS IS NOT THE DEFAULT SAVE METHOD. THIS FUNCTION IS NOT SUPPORTED IN WINDOWS STORE BUILDS.

```
LISTLOGICDRIVES()
```

```
public static string[] ListLogicDrives()
```

GET THE LIST OF THE AVAILABLE LOGIC DRIVES. THIS FUNCTION IS NOT SUPPORTED IN WINDOWS STORE BUILDS.

WARNING: NOT WIDELY TESTED ACROSS ALL PLATFORMS.

AES ENCRYPTION

```
#define USE_AES
```

DEFINE OR COMMENT THIS DIRECTIVE TO ALLOW AES ENCRYPTION.
THE AES ALGORITHM USES A FIXED LENGTH KEY, IT DOESN'T AFFECTS THE
XOR ALGORITHM DUE TO THAT ALGORITHM USES ANY KEY LENGTH.
DON'T FORGET TO SET YOUR OWN NEW KEY!
IMPORTANT: AES IS NOT SUPPORTED ON WINDOWS PHONE PLATFORMS.

STREAMINGASSETS INDEXER

THE STREAMINGASSETSINDEXER.CS SCRIPT WORKS AUTOMATICALLY WHILE IN
EDITOR AND GENERATES A FILE/SUB-DIRECTORY INDEX THAT CAN BE
NAVIGATED ONCE THE APPLICATION WAS BUILT FOR ANDROID OR WebGL.

THE INDEX FILE IS "FMWebGL" AND IS SAVED AUTOMATICALLY INTO YOUR
STREAMINGASSETS FOLDER TO BE INCLUDED WITHIN YOUR FINAL BUILD.
THE STREAMINGASSETSINDEXER.CS SCRIPT DETECTS MODIFICATIONS IN THE
FILE SYSTEM AND REGENERATES ALWAYS THE INDEX TO ENSURE THAT EVERY
EXISTING FILE OR SUB-DIRECTORY CAN BE DETECTED CORRECTLY.
THE STREAMINGASSETS FOLDER IS INACCESSIBLE IN ANDROID AND WebGL
BUILDS DUE TO IT IS NOT A REAL FOLDER, THAT IS THE MAIN REASON THE
INDEX WAS IMPLEMENTED.
THE INDEX INTERPRETATION IS FULLY INTEGRATED WITHIN THE [FileManagement](#)
CLASS.

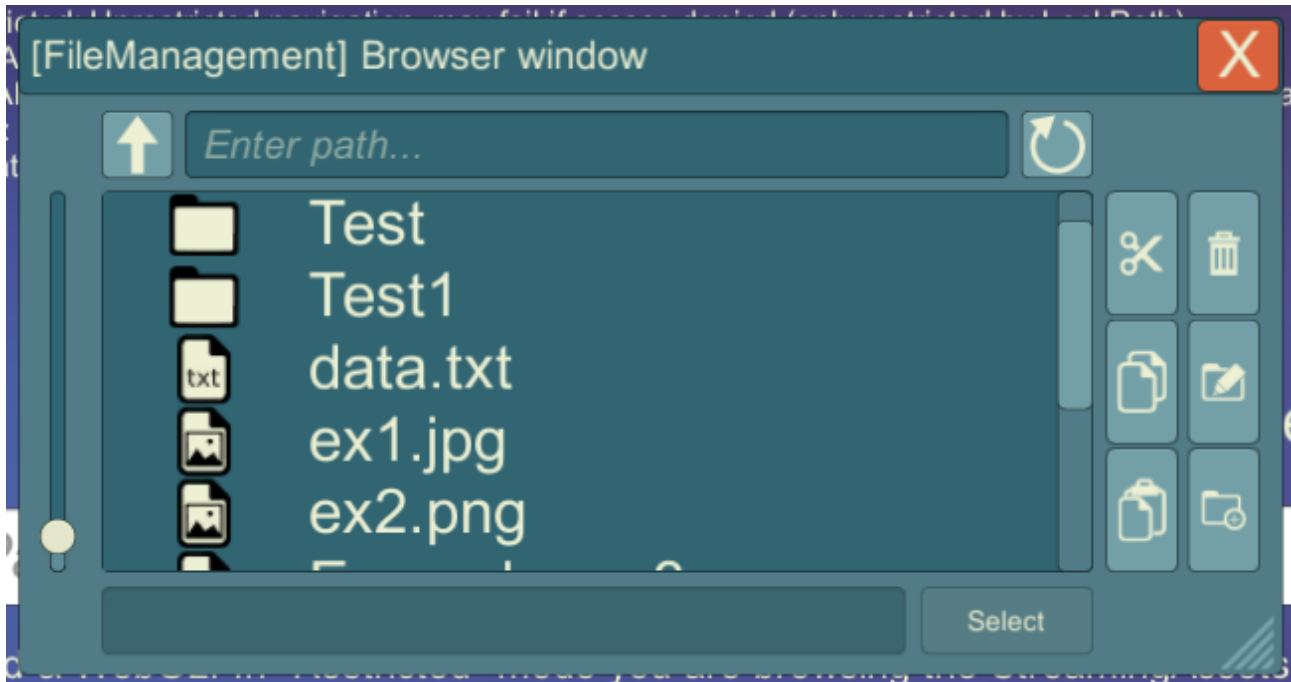
DO NOT MODIFY THE STREAMINGASSETS FOLDER OF YOUR FINAL BUILD, OR
THE INDEX WILL NOT MATCH THE CONTENT RESULTING IN ACCESSING
ERRORS. DO IT ALWAYS FROM UNITY EDITOR, THEN BUILD.

NOTE: THIS SCRIPT IS NOT INCLUDED IN YOUR FINAL BUILD.

FILEBROWSER PREFAB

THERE IS A PREFAB INCLUDED WITHIN THIS PACKAGE THAT ALLOWS YOU TO BROWSE THE FILE SYSTEM AND TO SELECT A FILE/FOLDER.

THE BROWSER IMPLEMENTS SOME USEFUL FUNCTIONALITY THAT ALLOWS IT TO BE USED WITHOUT PROGRAMMING SKILLS:

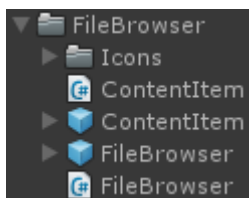


IT HAS BUILT IN CUT, COPY, PASTE, DELETE, RENAME AND NEW FODLER BUTTONS.

THE FILE BROWSER IS COMPLETELY DEVELOPED IN UNITY UI, SO YOU CAN CUSTOMIZE IT COMPLETELY AT YOUR WILL USING THE EDITOR TOOLS.

THE USE OF IMAGES ARE MAINTAINED AT MINIMUM TO ALLOW EXPORT EASILY. THE ONLY IMAGE RESOURCES THAT USES THE PREFAB ARE CONTAINED IN THE "ICONS" FOLDER.

THIS IS WHAT YOU NEED:



THE SCRIPTS USED IN THIS PREFAB ARE DEPENDENT OF THE [FileManagement](#) CLASS, SO YOU WILL NEED IT TO BE IMPORTED TOO.

THE "ICONS" FOLDER CONTAINS THE IMAGE RESOURCES.

THE [FileBrowser](#) DOESN'T OPENS THE FILE, IT JUST RETURNS THE CORRECT PATH TO BE TREATED IN THE MAIN APPLICATION WHEN IT CLOSSES.

THE WAY YOU CREATE A `FileBrowser` IS AS FOLLOWS:

```
public GameObject fileBrowser; // Drag the FileBrowser prefab here (in the editor).

// Create a FileBrowser window (with default options):
public void OpenFileBrowser()
{
    GameObject browserInstance = GameObject.Instantiate(fileBrowser);
    browserInstance.GetComponent<FileBrowser>().SetBrowserWindow(OnPathSelected);
}

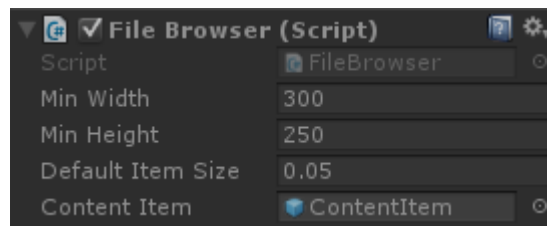
// You should use this function signature in order to receive properly:
void OnPathSelected(string path)
{
    // Do something with the returned path.
}
```

THE `FileBrowser` DESTROYS ITSELF ONCE A FILE IS SELECTED OR THE WINDOW IS CLOSED. YOU CAN ACCESS SOME PUBLIC FUNCTIONS THROUGH THE `browserInstance` VARIABLE.

YOU CAN DOUBLE-CLICK OR DOUBLE-TAP THE ITEMS TO NAVIGATE AND CHOSE.

FILEBROWSER.CS

THERE ARE A FEW PARAMETERS THAT CAN BE CUSTOMIZED DIRECTLY FROM EDITOR IN THE `FILEBROWSER.CS` SCRIPT:



`int minWidth`: THE MINIMUM WIDTH OF THE WINDOW WHEN RESIZING (PIXELS).

`int minHeight`: THE MINIMUM HEIGHT OF THE WINDOW WHEN RESIZING (PIXELS).

`float defaultItemSize`: THE DEFAULT SIZE FOR ITEMS IF THE SIZE SLIDER IS DISABLED OR DELETED (PERCENTAGE OF THE CANVAS HEIGHT IN UNITS).

`public GameObject ContentItem`: THIS IS THE REFERENCE TO THE ITEM PREFAB.

HERE IS THE DEFINITION OF THE INTERFACE USED TO SET THE `FileBrowser` INSTANCE (YOU WILL FIND IT IN `FILEBROWSER.CS`):

```
public void SetBrowserWindow(OnPathSelected selectionReturn, string iniPath = "",
bool fullPath = false, string selectionMode = "F", bool save = false, string lockPath = "")
```

THE `OnPathSelected` DELEGATE HAS A SPECIFIC SIGNATURE IN ORDER TO BE

CALLLED PROPERLY: `void MyFunction(string)`. THE `ReturnSelectedFile()` FUNCTION EXECUTES THE DELEGATE, RETURNING THE SELECTED PATH/FILE AND CLOSES THE WINDOWS.

THE `iniPath` ARGUMENT SETS THE FIRST FOLDER TO BE SHOWN WHEN THE WINDOW BECOMES VISIBLE.

THE `fullPath` ARGUMENT ALLOWS THE BROWSER TO NAVIGATE OUTSIDE THE UNIFIED `PERSISTENTDATA+STREAMINGASSETS` PATH WITHOUT RESTRICTIONS.

THE `selectionMode` ARGUMENT ALLOWS THE BROWSER TO SELECT FILES OR FOLDERS DEPENDING ON ITS VALUE: "F" FOR FILES, "D" FOR DIRECTORIES.

THE `save` ARGUMENT ALLOWS THE `InputField` OF THE SELECTED ITEM TO ALLOW WRITING A CUSTOM FILE OR FOLDER NAME.

THE `lockPath` ARGUMENT FORCES THE BROWSER TO STAY ALWAYS INTO THAT DIRECTORY AND SUB-DIRECTORIES NOT ALLOWING FURTHER NAVIGATION (NO MATTER WHAT NAVIGATION MODE WAS SELECTED).

IMPORTANT: THIS SCRIPT ASSUMES THAT IS ATTACHED TO THE ROOT `GameObject` OF THE PREFAB (USES THE `RectTransform` OF THE CANVAS).

THERE ARE SOME OTHER VERY USEFUL FUNCTIONS:

CUT: THIS BUTTON CALLS `Cut()` REMEMBERING THE FILE/FOLDER TO CUT. IT WILL SHOW A WARNING A MESSAGE IF THE FILE/FOLDER IS READ ONLY.

COPY: THIS BUTTON CALLS `Copy()` REMEMBERING THE FILE/FOLDER TO COPY.

PASTE: THIS BUTTON CALLS `Paste()` COPYING OR MOVING THE PREVIOUSLY SELECTED FILE OR FOLDER AND ALL ITS CONTENT.

DELETE: THIS BUTTON CALLS `PromptDeleteSelection()` ALLOWING FILE/FOLDER DELETION. IT WILL SHOW A WARNING A MESSAGE IF THE FILE/FOLDER IS READ ONLY.

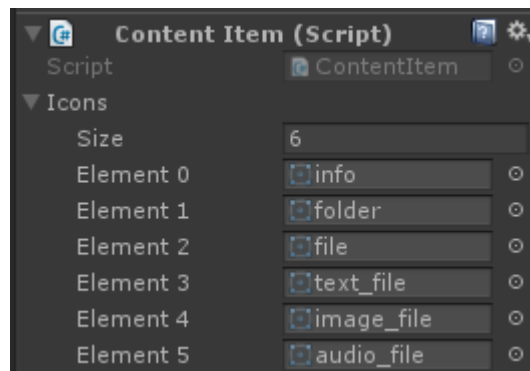
RENAME: THIS BUTTON CALLS `PromptForRename()` ALLOWING RENAME A FILE OR FOLDER. IT WILL SHOW A WARNING A MESSAGE IF THE FILE/FOLDER IS READ ONLY.

NEW FOLDER: THIS BUTTON CALLS `PromptNewFolderName()` ALLOWING CREATE A NEW FOLDER. YOU CAN CREATE NESTED FOLDERS USING A PATH NOTATION ("`Folder1/Folder2`").

CONTENTITEM.CS

THIS SCRIPT CONTROLS THE BEHAVIOR OF THE RENDERED ITEMS.

YOU CAN MODIFY THE ICONS SHOWN IN ITEMS FROM HERE (IN THE PREFAB) ADDING OR REPLACING THEM IN THE icons ARRAY OF [Sprite](#) ELEMENTS.



THE ICONS RENDERED DEPENDING ON THE FILE EXTENSION ARE DETERMINED BY THE `SetItem` METHOD IN THE `CONTENTITEM.CS` SCRIPT. FEEL FREE TO MODIFY AND ADD YOUR OWN FILE EXTENSIONS AND ICONS.

CUSTOMIZE THE BROWSER WINDOW

THE FILE BROWSER IS COMPLETELY BUILT WITH THE UNITY UI, SO YOU CAN USE THE STANDARD TOOLS TO CUSTOMIZE THE WINDOW AT YOUR WILL.

BEFORE START CUSTOMIZING THE PREFAB, PLEASE MAKE A COPY WITH `CTRL+D` AND RENAME AS NEEDED. DUPLICATE THE `CONTENTITEM` PREFAB TOO, AND ONCE RENAMED, ASSIGN ITS REFERENCE TO THE `ContentItem` VARIABLE INTO YOUR NEW FILE BROWSER PREFAB.

THERE ARE NO SCRIPTS ATTACHED TO THE ELEMENTS OF THE BROWSER WINDOW, EVERYTHING IS CONNECTED FROM `FILEBROWSER.CS`.

YOU JUST HAVE TO KEEP IN MIND TO USE SOME SPECIAL NAMES IN ITS HIERARCHY TO ALLOW THE `FILEBROWSER.CS` SCRIPT TO FIND THE ELEMENTS IT USES TO WORK PROPERLY.

THOSE ELEMENTS ARE:

BrowserWindow: THIS IS MAIN CONTAINER PANEL. EVERYTHING IS CONTAINED BY THIS ELEMENT.

InputCurrentPath: THIS [InputField](#) CONTAINS THE CURRENT DIRECTORY BEING RENDERED.

ContentWindow>VIEWPORT>CONTENT: THIS IS THE MAIN CONTAINER OF THE RENDERED ITEMS. THE `CONTENTITEM` PREFABS ARE ADDED AUTOMATICALLY TO ITS [Transform](#). IT HAS A [VerticallyLayoutGroup](#) COMPONENT TO ARRANGE ITEMS.

ContentWindow: THIS [ScrollRect](#) IS THE LIST CONTROLLER OF RENDERED ITEMS.

InputSelection: THIS [InputField](#) CONTAINS THE NAME OF THE SELECTED ITEM. WHEN IN save MODE, THIS [InputField](#) CAN BE EDITED TO SET A CUSTOM FILE OR FOLDER NAME.

ButtonSelect: THIS IS THE SELECTION **Button**, IT CAN BE ENABLED OR DISABLED DEPENDING ON THE CURRENT SELECTION.

ButtonSelect>Text: THE **Text** OF THE **ButtonSelect** CHANGES DEPENDING ON THE ACTION THAT CAN BE EXECUTED: "Select", "Open" OR "Save".

SizeSlider: THIS **Slider** ALLOWS MODIFYING THE ITEM SIZE DYNAMICALLY. CAN BE DELETED IF NOT USED.

Confirmation: THIS IS THE CONFIRMATION POP-UP WITH A **Label** AND TWO BUTTONS: **ButtonOk** AND **ButtonCancel**.

NewItem: THIS IS THE TEXT INPUT POP-UP WITH A **Label**, AN **InputField** CALLED **InputNewItem** AND TWO BUTTONS: **ButtonOk** AND **ButtonCancel**.

ErrorMessage: THIS IS THE MESSAGE POP-UP WITH A **Label**, AN **Image** AND A BUTTON: **ButtonOk**.

EXCEPTING **ButtonSelect** BUTTON, EVERY BUTTON OR INTERACTIVE ELEMENT HAS AN EVENT THAT POINTS TO A METHOD CONTAINED IN **FileBrowser.cs**.

KNOWN ISSUES

- THERE IS NO DIRECT COMPATIBILITY WITH THE LEGACY INTERFACES USED FOR READING AND WRITING COOKIES IN WEB BROWSERS. NEVERTHELESS YOU WILL FIND THOSE INTERFACES AS `private` IN THE `FileManagement` CLASS SO YOU CAN STILL USING THEM IF NEEDED (WEBGL ONLY).
- TO GRANT SDCARD ACCESS IN ANDROID YOU MUST ENABLE THE OPTION: BUILDSETTINGS>PLAYERSETTINGS>OTHERSETTINGS>WRITEACCESS = EXTERNAL (SDCARD).

CONTACT

IF YOU NEED SOME NEW INTERFACES OR IF YOU FIND SOME ERRORS IN THIS DOCUMENTATION OR THE APPLICATION, DON'T HESITATE ON SEND ME AN EMAIL: [JMONSUAREZ@GMAIL.COM](mailto:jmonsuarez@gmail.com)

I YOU FIND THAT THE TEST VERSION IS NOT THE SAME VERSION THAT YOU DOWNLOADED FROM THE ASSETSTORE, PLEASE SEND ME YOUR INVOICE NUMBER AND I WILL SEND YOU BACK THE LAST FILEMANAGEMENT VERSION (NEW VERSION NORMALLY IS INTO APPROVAL PROCESS).

PLEASE, ONCE YOU HAVE TESTED THIS PRODUCT, TAKE A MINUTE OF YOUR TIME TO WRITE A GOOD REVIEW IN THE UNITY ASSET STORE, SO YOU WILL HELP TO IMPROVE THIS PRODUCT:

[HTTPS://WWW.ASSETSTORE.UNITY3D.COM/EN/#!/CONTENT/67183](https://www.assetstore.unity3d.com/en/#!/content/67183)

THANKS.