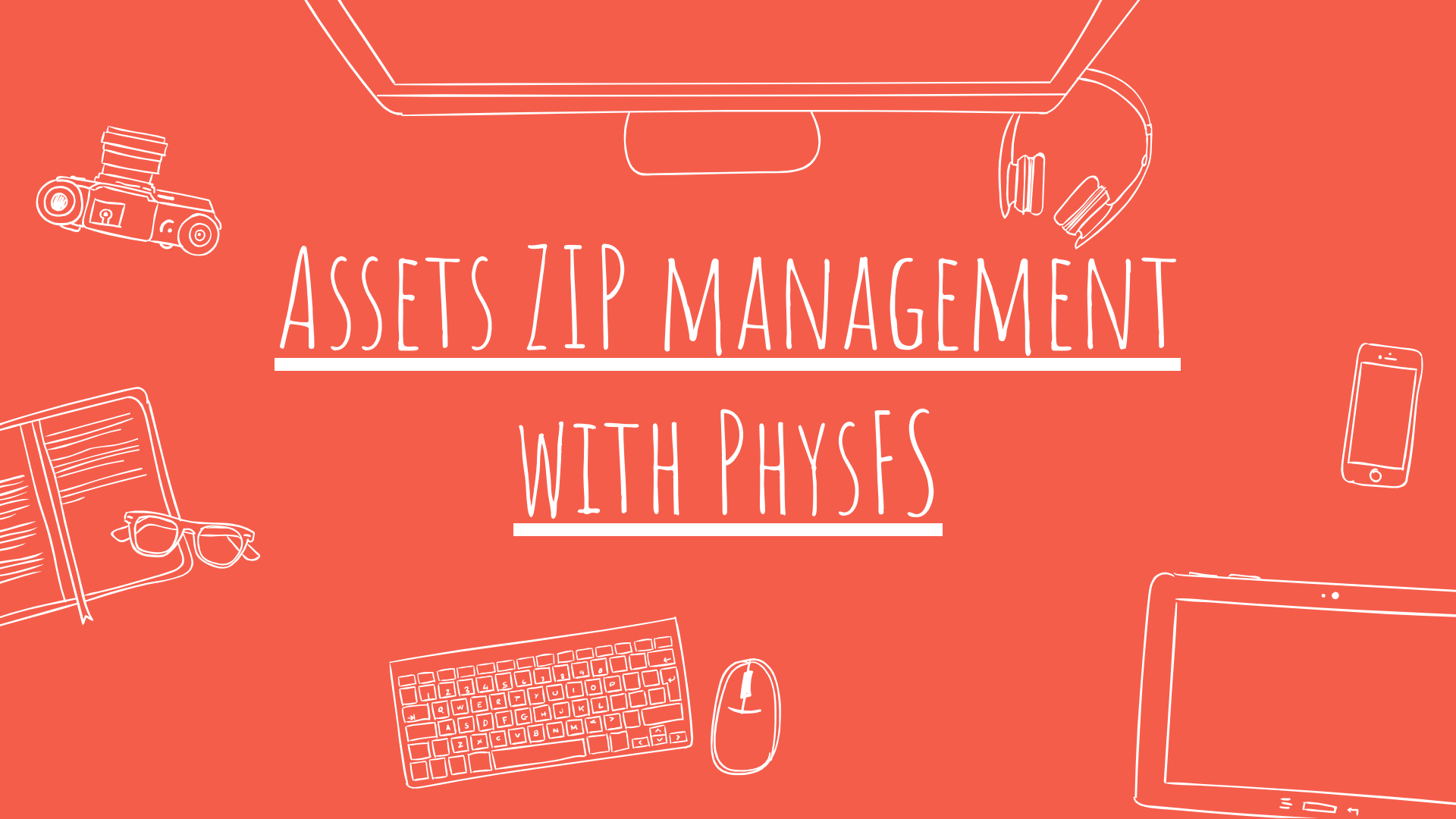
Assets ZIP management with PhysFS



INDEX







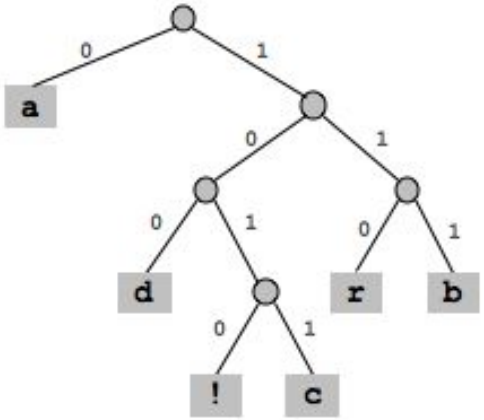
1.

What is ZIP RAR?. How do they work? What are compression algorithms?

1.1 - WHAT IS A COMPRESSION ALGORITHM?

Is the process of encoding information using fewer bits than the original representation.

1 char = 1 byte =8 bits



char	encoding
a	0
b	111
c	1011
d	100
r	110
!	1010

1.1 - WHAT IS A COMPRESSION ALGORITHM?

TWO TYPES OF COMPRESSION ALGORITHMS

lossy compression algorithms

Low-compression (high quality) JPEG

High-compression (low quality) JPEG

IMAGE:

- JPEG
- GIF

VIDEO:

- Mp4
- Motion JPEG

AUDIO:

- Mp3
- OGG

lossless compression algorithms

IMAGE:

- PNG
- RAW

VIDEO:

- AVI
- MOV

AUDIO:

- FLAC
- MPEG-4

RAR, ZIP

1.2 - WHAT IS ZIP AND RAR?

ZIP:

- Is a lossless compression format.
- Uses the lz77 compression algorithm with huffman encoding.
- Compress each file independently of the rest of the files.
- Has a lower compression rate than RAR compression.

1.2 - WHAT IS ZIP AND RAR?

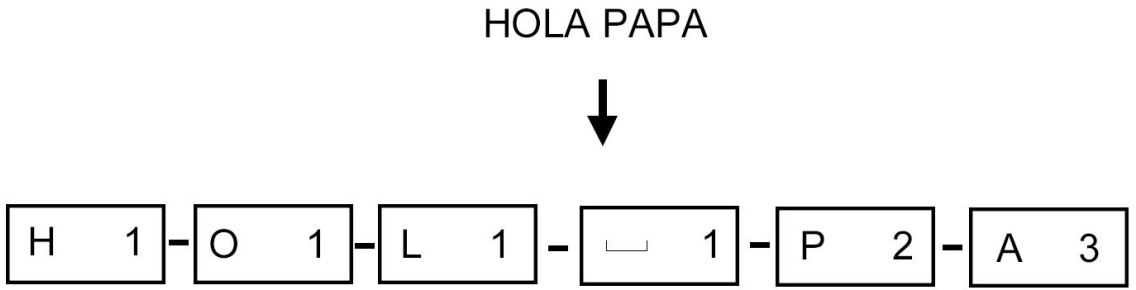
RAR:

- Is a lossless compression format.
- Uses the LZSS compression algorithm which is based on LZ77.
- Compress all files together.
- Has a higher compression rate than ZIP.

1.3 - HOW COMPRESSION ALGORITHMS WORK?

Huffman encoding:

- We create a list ordering the characters according to their frequency in the message.

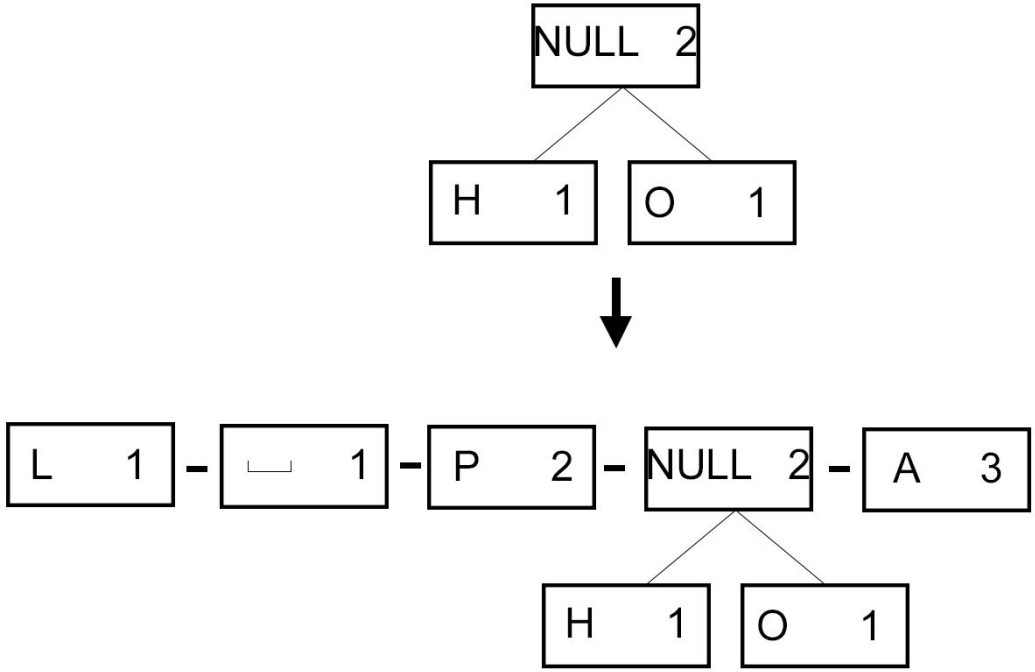


1.3 - HOW COMPRESSION ALGORITHMS WORK?

Huffman encoding:

- We take the first two elements of the list and create a parent node.

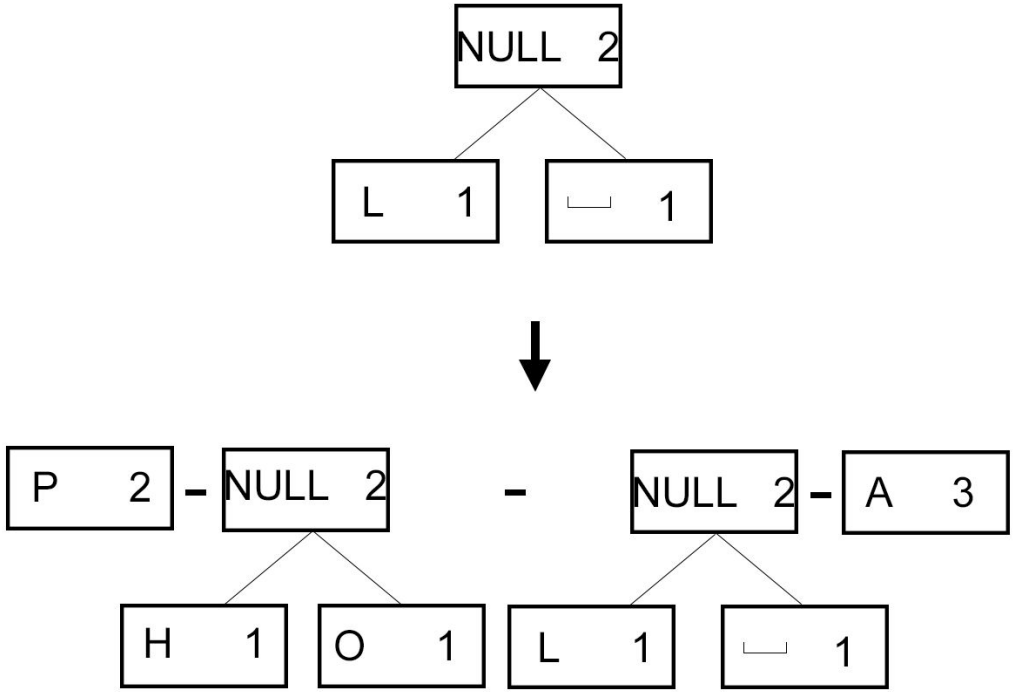
- Then we add it back to the list.



1.3 - HOW COMPRESSION ALGORITHMS WORK?

Huffman encoding:

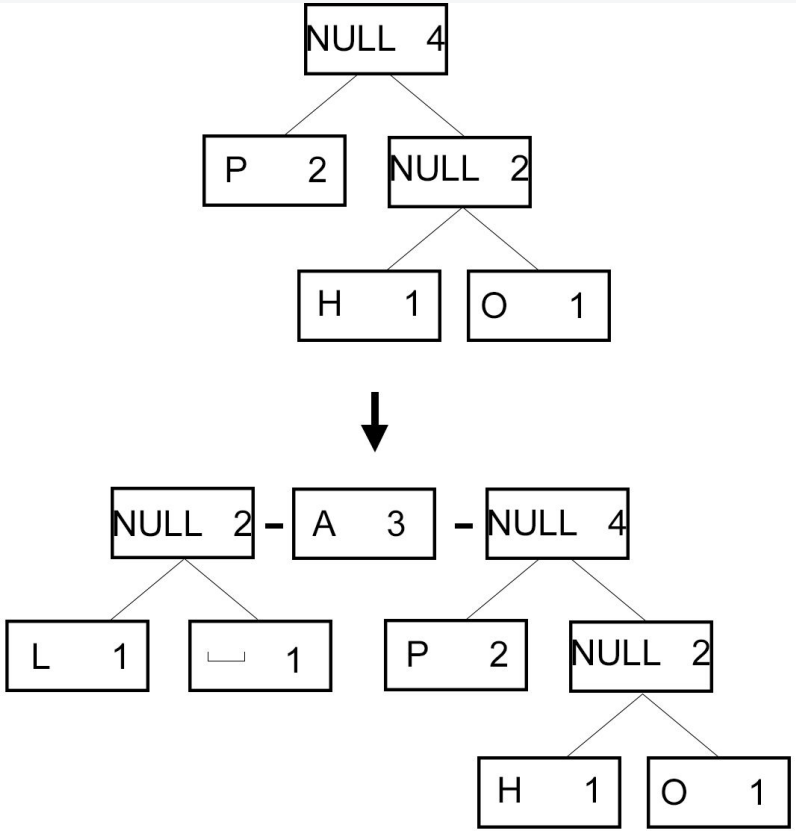
- We repeat this process until we have a binary tree.



1.3 - HOW COMPRESSION ALGORITHMS WORK?

Huffman encoding:

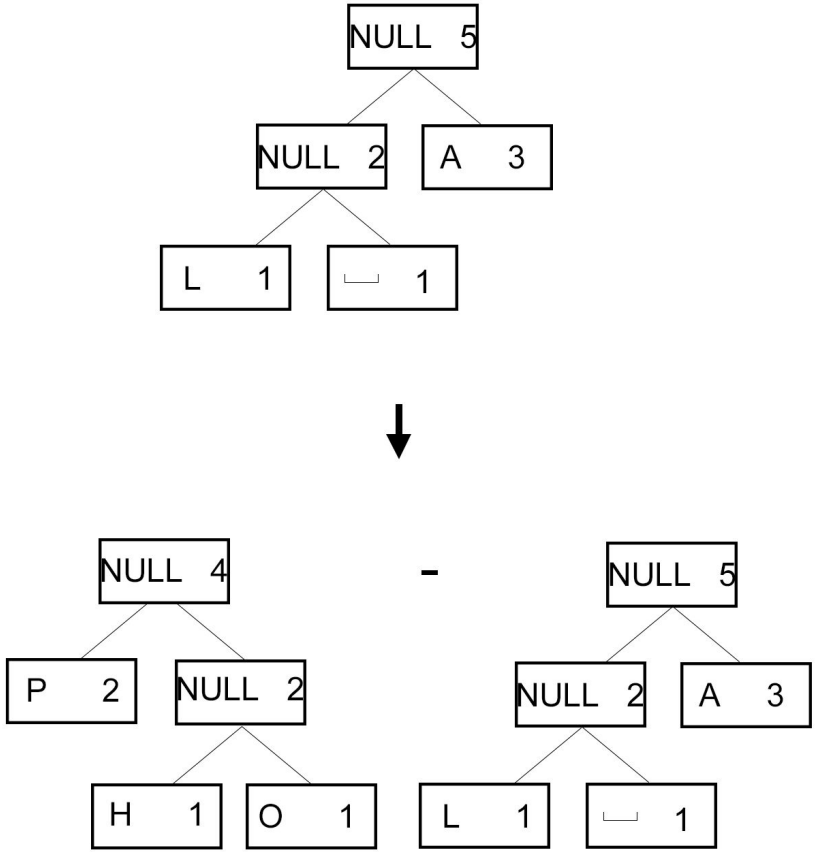
- We repeat this process until we have a binary tree.



1.3 - HOW COMPRESSION ALGORITHMS WORK?

Huffman encoding:

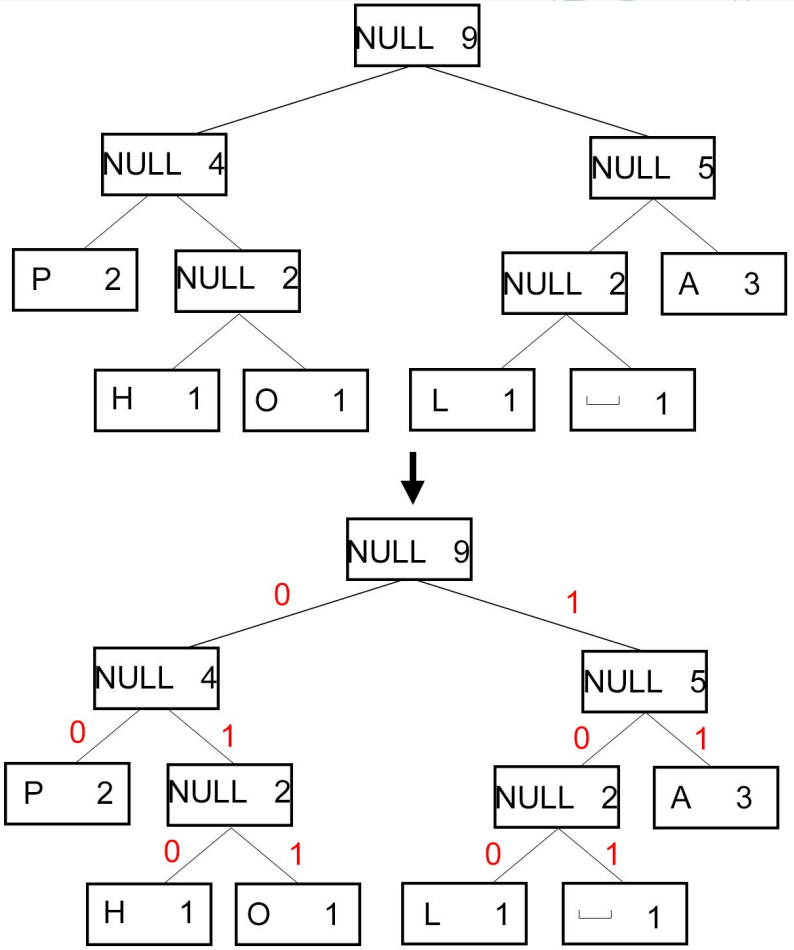
- We repeat this process until we have a binary tree.



1.3 - HOW COMPRESSION ALGORITHMS WORK?

Huffman encoding:

- When we finish the process we will put a 0 on the left and a 1 on the right of each branch to identify each character.



1.3 - HOW COMPRESSION ALGORITHMS WORK?

Huffman encoding:

- We create a table with each character and its code from the binary tree.

- We have reduced the weight of the message from 72 bits to 22 bits.

H	010
O	011
L	100
A	11
␣	101
P	00

HOLA PAPA

01001000 01001111 01001100 01000001 00100000 01010000
H O L A ␣ P
01000001 01010000 01000001 -- 72 bits.
A P A

010 011 100 11 101 00 11 00 11 -- 22 bits
H O L A ␣ P A P A

1.3 - HOW COMPRESSION ALGORITHMS WORK?

LZ77:

- We create the table that will be used to encode and decode the information.

1 next character

2 spaces back

3 number of spaces to copy

ABRACADABRA

1	2	3
A	0	0

1.3 - HOW COMPRESSION ALGORITHMS WORK?

LZ77:

- We are checking if the character that comes is already in the table.

1 next character

2 spaces back

3 number of spaces to copy

ABRACADABRA

	1	2	3
A	0	0	
B	0	0	

1.3 - HOW COMPRESSION ALGORITHMS WORK?

LZ77:

1 next character

2 spaces back

3 number of spaces to copy

ABRACADABRA

1	2	3
A	0	0
B	0	0
R	0	0

1.3 - HOW COMPRESSION ALGORITHMS WORK?

LZ77:

ABRACADABRA

1 next character

2 spaces back

3 number of spaces to copy

1	2	3
A	0	0
B	0	0
R	0	0
C	3	1

1.3 - HOW COMPRESSION ALGORITHMS WORK?

LZ77:

ABRACADABRA

1 next character

2 spaces back

3 number of spaces to copy

1	2	3
A	0	0
B	0	0
R	0	0
C	3	1
D	2	1

1.3 - HOW COMPRESSION ALGORITHMS WORK?

LZ77:

- Finally we get this table, it will be all the information we need to later decode the message.

1 next character

2 spaces back

3 number of spaces to copy

ABRACADABRA

1	2	3
A	0	0
B	0	0
R	0	0
C	3	1
D	2	1
NULL	7	4







2.

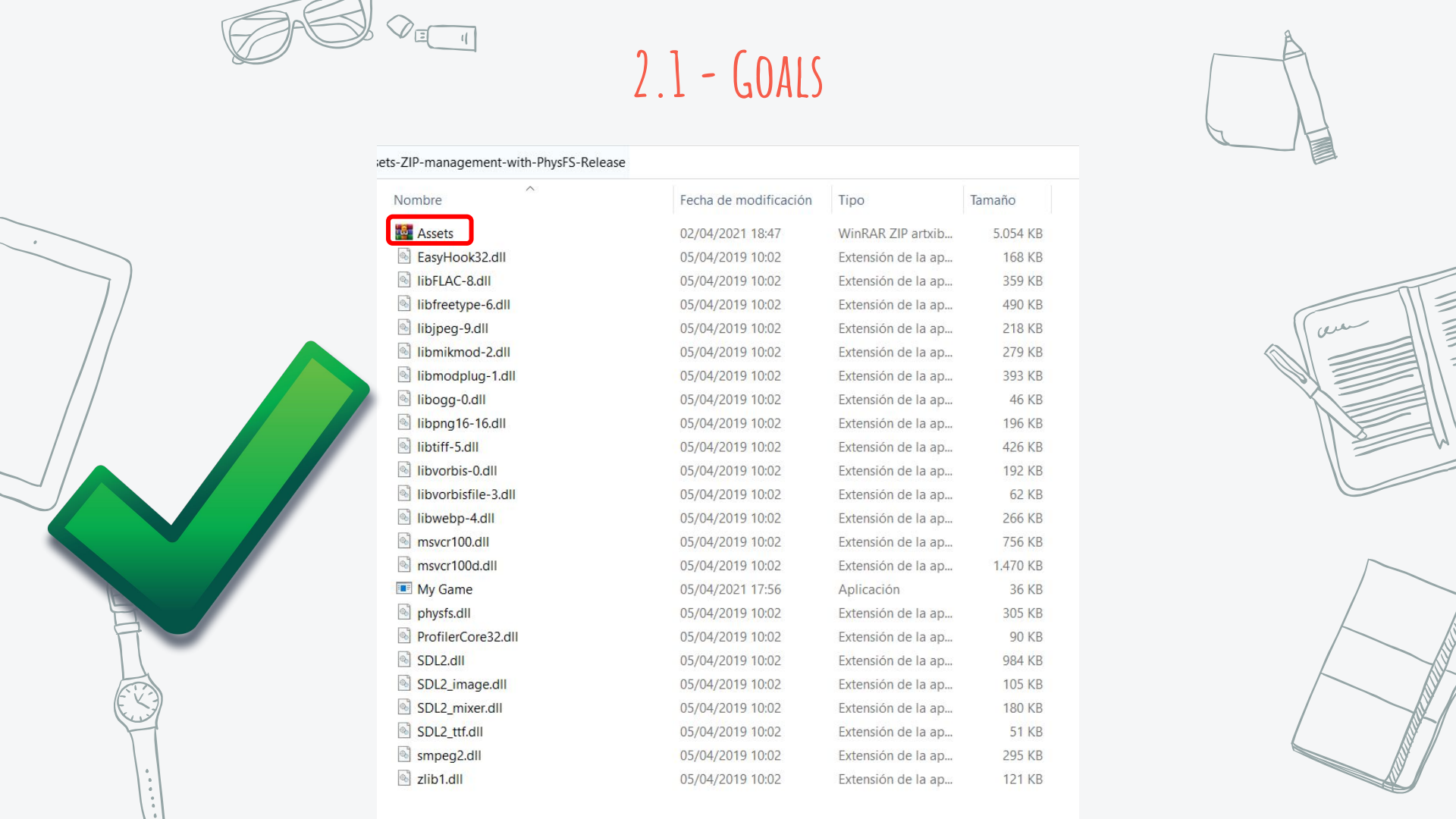
What is physfs?, Advantages of using it, objectives with the library.

2.1 - Goals

ets-ZIP-management-with-PhysFS-Release

Nombre	Fecha de modificación	Tipo	Tamaño
Assets	06/04/2021 16:01	Carpeta de archivos	
EasyHook32.dll	05/04/2019 10:02	Extensión de la ap...	168 KB
libFLAC-8.dll	05/04/2019 10:02	Extensión de la ap...	359 KB
libfreetype-6.dll	05/04/2019 10:02	Extensión de la ap...	490 KB
libjpeg-9.dll	05/04/2019 10:02	Extensión de la ap...	218 KB
libmikmod-2.dll	05/04/2019 10:02	Extensión de la ap...	279 KB
libmodplug-1.dll	05/04/2019 10:02	Extensión de la ap...	393 KB
libogg-0.dll	05/04/2019 10:02	Extensión de la ap...	46 KB
libpng16-16.dll	05/04/2019 10:02	Extensión de la ap...	196 KB
libtiff-5.dll	05/04/2019 10:02	Extensión de la ap...	426 KB
libvorbis-0.dll	05/04/2019 10:02	Extensión de la ap...	192 KB
libvorbisfile-3.dll	05/04/2019 10:02	Extensión de la ap...	62 KB
libwebp-4.dll	05/04/2019 10:02	Extensión de la ap...	266 KB
msvcr100.dll	05/04/2019 10:02	Extensión de la ap...	756 KB
msvcr100d.dll	05/04/2019 10:02	Extensión de la ap...	1.470 KB
My Game	05/04/2021 17:56	Aplicación	36 KB
physfs.dll	05/04/2019 10:02	Extensión de la ap...	305 KB
ProfilerCore32.dll	05/04/2019 10:02	Extensión de la ap...	90 KB
SDL2.dll	05/04/2019 10:02	Extensión de la ap...	984 KB
SDL2_image.dll	05/04/2019 10:02	Extensión de la ap...	105 KB
SDL2_mixer.dll	05/04/2019 10:02	Extensión de la ap...	180 KB
SDL2_ttf.dll	05/04/2019 10:02	Extensión de la ap...	51 KB
smpeg2.dll	05/04/2019 10:02	Extensión de la ap...	295 KB
zlib1.dll	05/04/2019 10:02	Extensión de la ap...	121 KB

2.1 - GOALS



ets-ZIP-management-with-PhysFS-Release

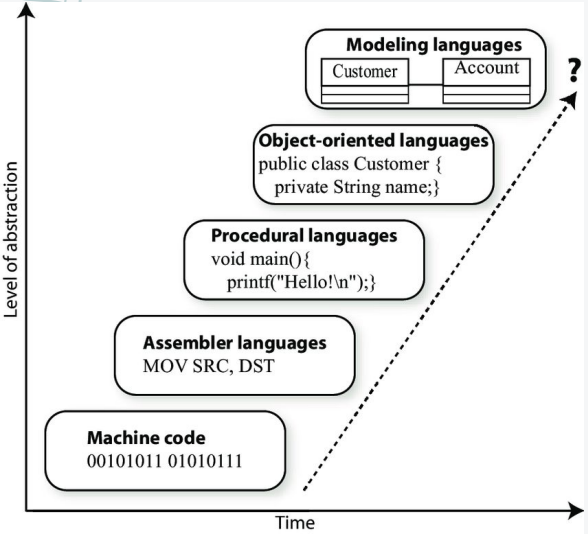
Nombre	Fecha de modificación	Tipo	Tamaño
Assets	02/04/2021 18:47	WinRAR ZIP artxib...	5.054 KB
EasyHook32.dll	05/04/2019 10:02	Extensión de la ap...	168 KB
libFLAC-8.dll	05/04/2019 10:02	Extensión de la ap...	359 KB
libfreetype-6.dll	05/04/2019 10:02	Extensión de la ap...	490 KB
libjpeg-9.dll	05/04/2019 10:02	Extensión de la ap...	218 KB
libmikmod-2.dll	05/04/2019 10:02	Extensión de la ap...	279 KB
libmodplug-1.dll	05/04/2019 10:02	Extensión de la ap...	393 KB
libogg-0.dll	05/04/2019 10:02	Extensión de la ap...	46 KB
libpng16-16.dll	05/04/2019 10:02	Extensión de la ap...	196 KB
libtiff-5.dll	05/04/2019 10:02	Extensión de la ap...	426 KB
libvorbis-0.dll	05/04/2019 10:02	Extensión de la ap...	192 KB
libvorbisfile-3.dll	05/04/2019 10:02	Extensión de la ap...	62 KB
libwebp-4.dll	05/04/2019 10:02	Extensión de la ap...	266 KB
msvcr100.dll	05/04/2019 10:02	Extensión de la ap...	756 KB
msvcr100d.dll	05/04/2019 10:02	Extensión de la ap...	1.470 KB
My Game	05/04/2021 17:56	Aplicación	36 KB
physfs.dll	05/04/2019 10:02	Extensión de la ap...	305 KB
ProfilerCore32.dll	05/04/2019 10:02	Extensión de la ap...	90 KB
SDL2.dll	05/04/2019 10:02	Extensión de la ap...	984 KB
SDL2_image.dll	05/04/2019 10:02	Extensión de la ap...	105 KB
SDL2_mixer.dll	05/04/2019 10:02	Extensión de la ap...	180 KB
SDL2_ttf.dll	05/04/2019 10:02	Extensión de la ap...	51 KB
smpeg2.dll	05/04/2019 10:02	Extensión de la ap...	295 KB
zlib1.dll	05/04/2019 10:02	Extensión de la ap...	121 KB

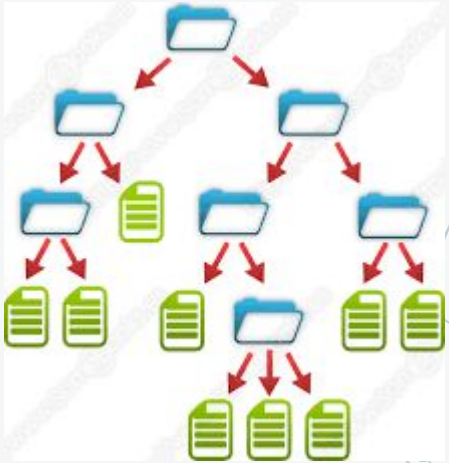
2.2 What is Phys Fs?

PhysFs is a library that allows us to access files in an abstract way.

This will be very useful for video games.

Thanks to the flexibility of the library we will be able to work with all types of files, including Zip.





-Abstract: Organize our data into files, directories, and other constructs, and manipulate them in various ways. It's not necessary to figure out the exact location on disk.

2.3 Why use Phys Fs?

Advantage

- Security, the library will not allow

  reading or writing outside of the given directory.

- Flexibility, we can use Zip files as a directory.

- Assets take up less space in our project.

Drawback

- We must introduce a new library.

- Asset loading will take longer







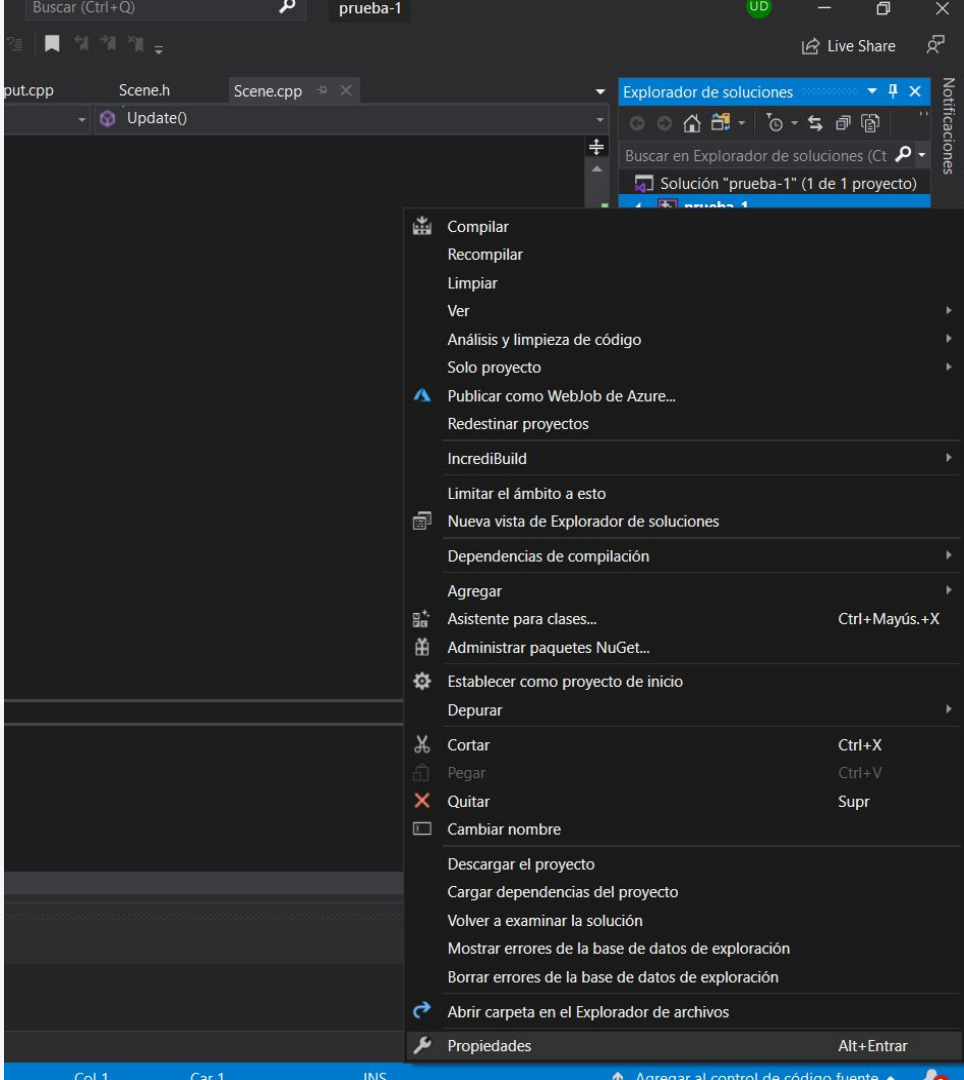
3.

How do we implement it in the project?.

3.1 - How to implement it

Step 1:

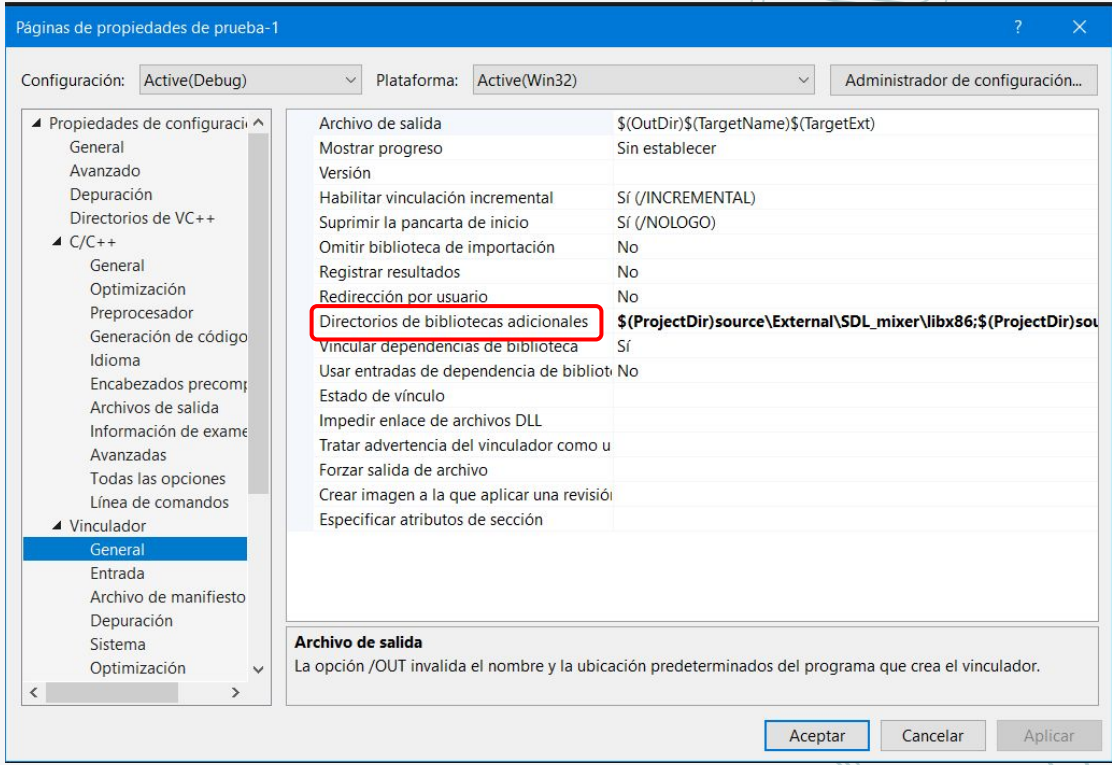
-Open the project properties to edit the linker.



3.1 - How to implement it

Step 2:

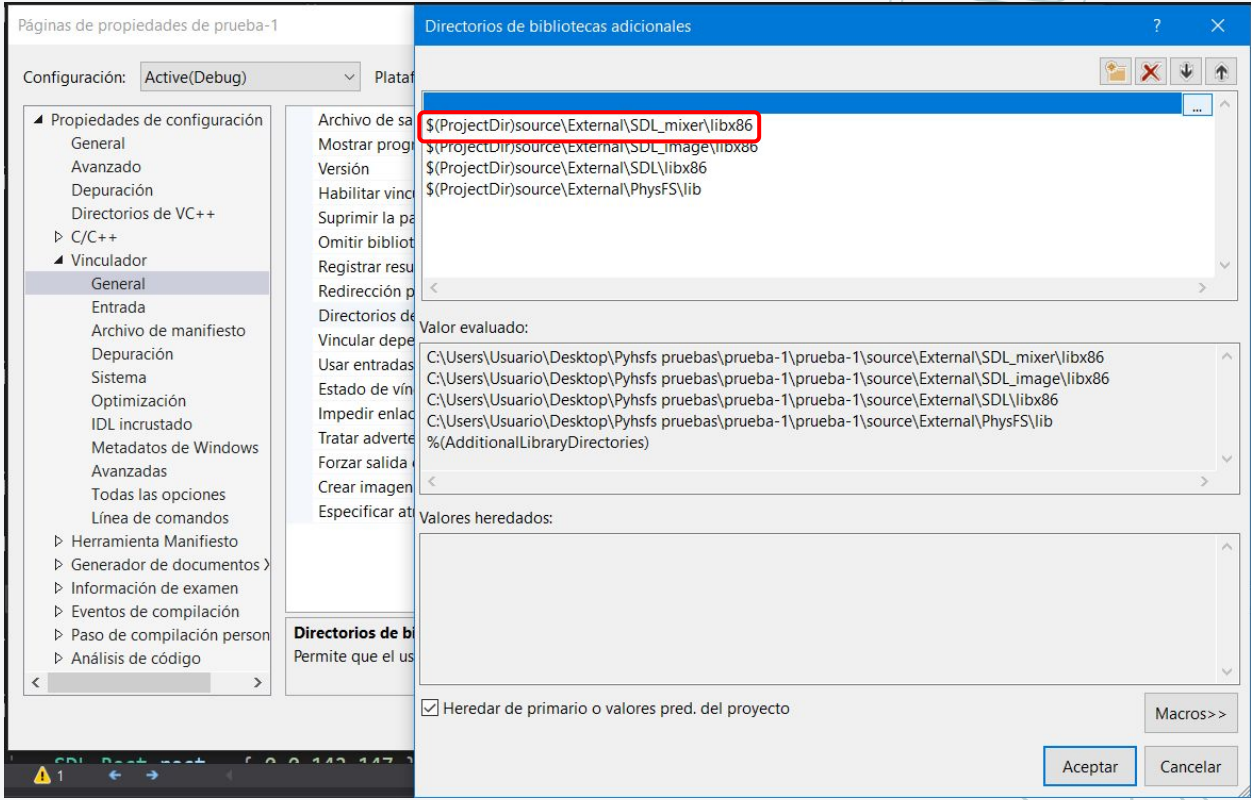
-Open the linker tab and in the general section, edit the "Directories of additional libraries".



3.1 - How to implement it

Step 3:

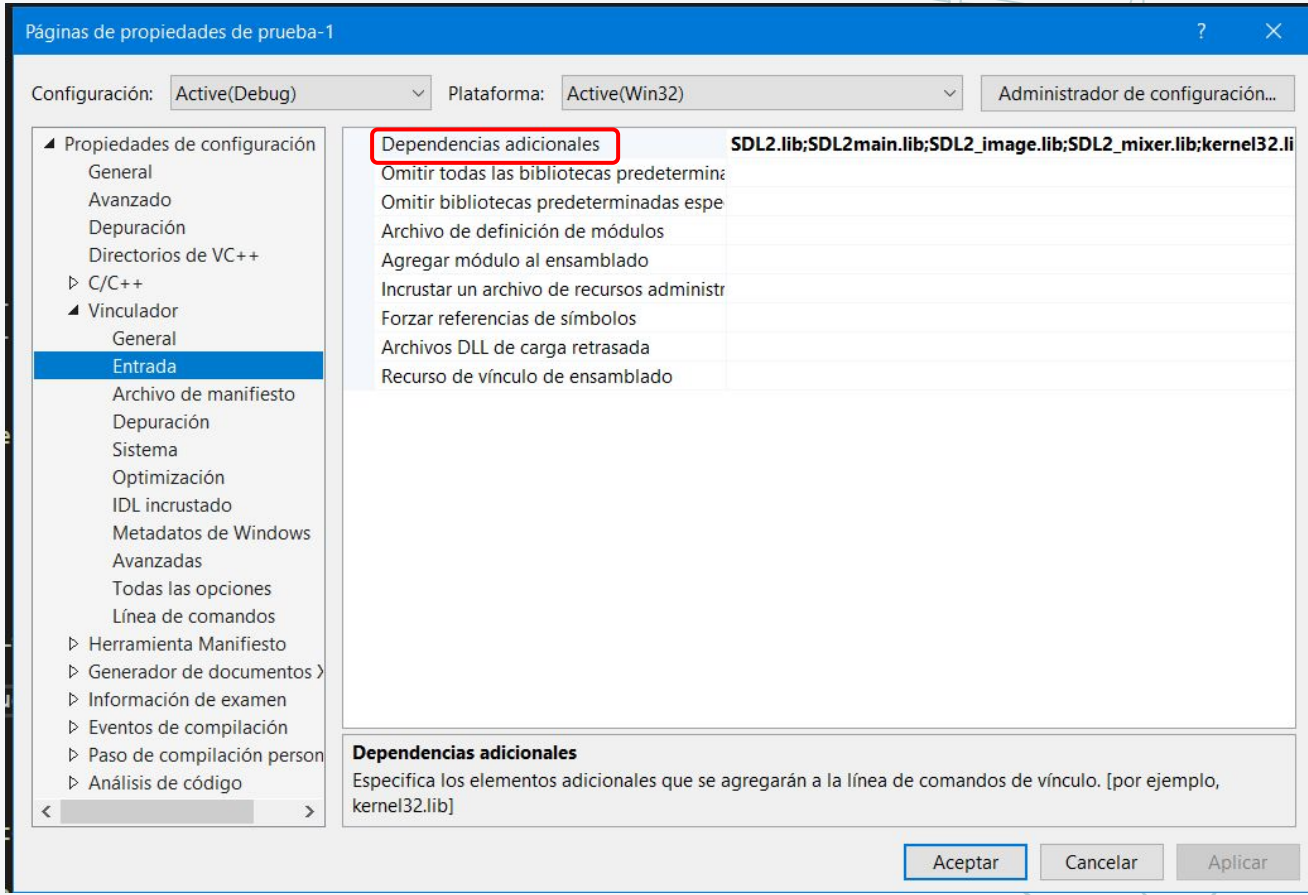
-Indicate in which project folder the different libraries are located.



3.1- How to implement it

Step 4:

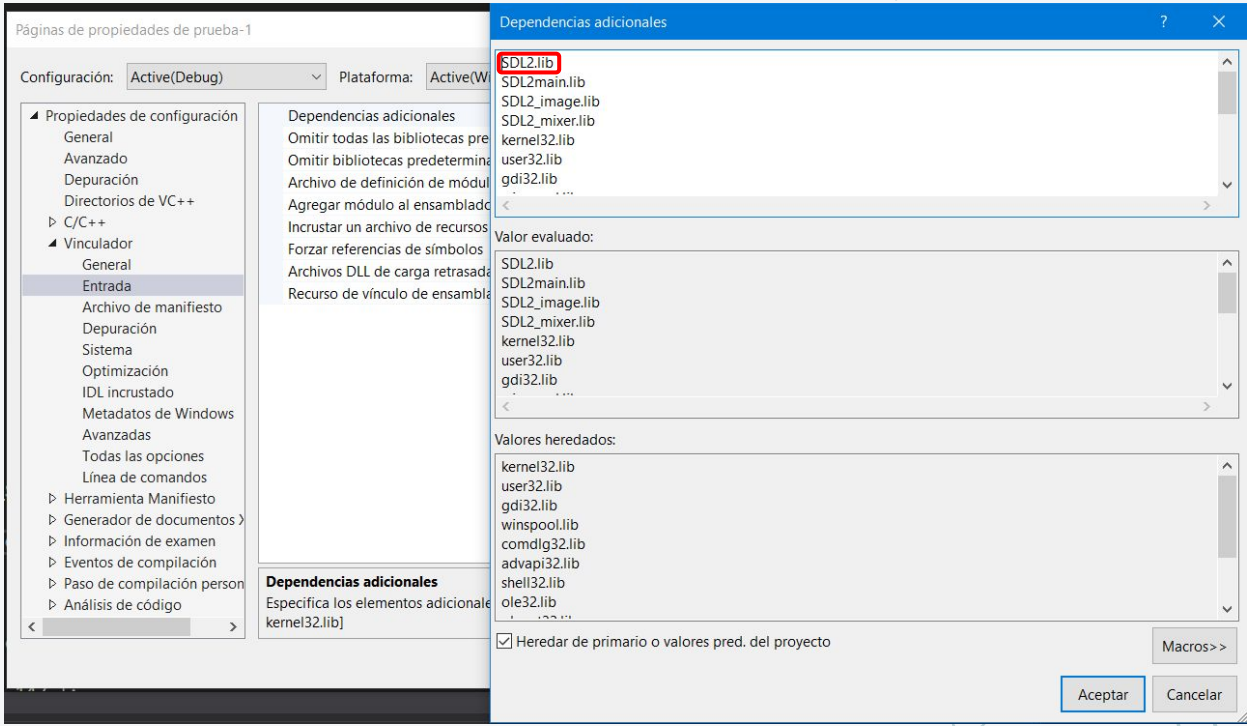
-In the input section, edit the section of "additional dependencies".



3.1 - How to implement it

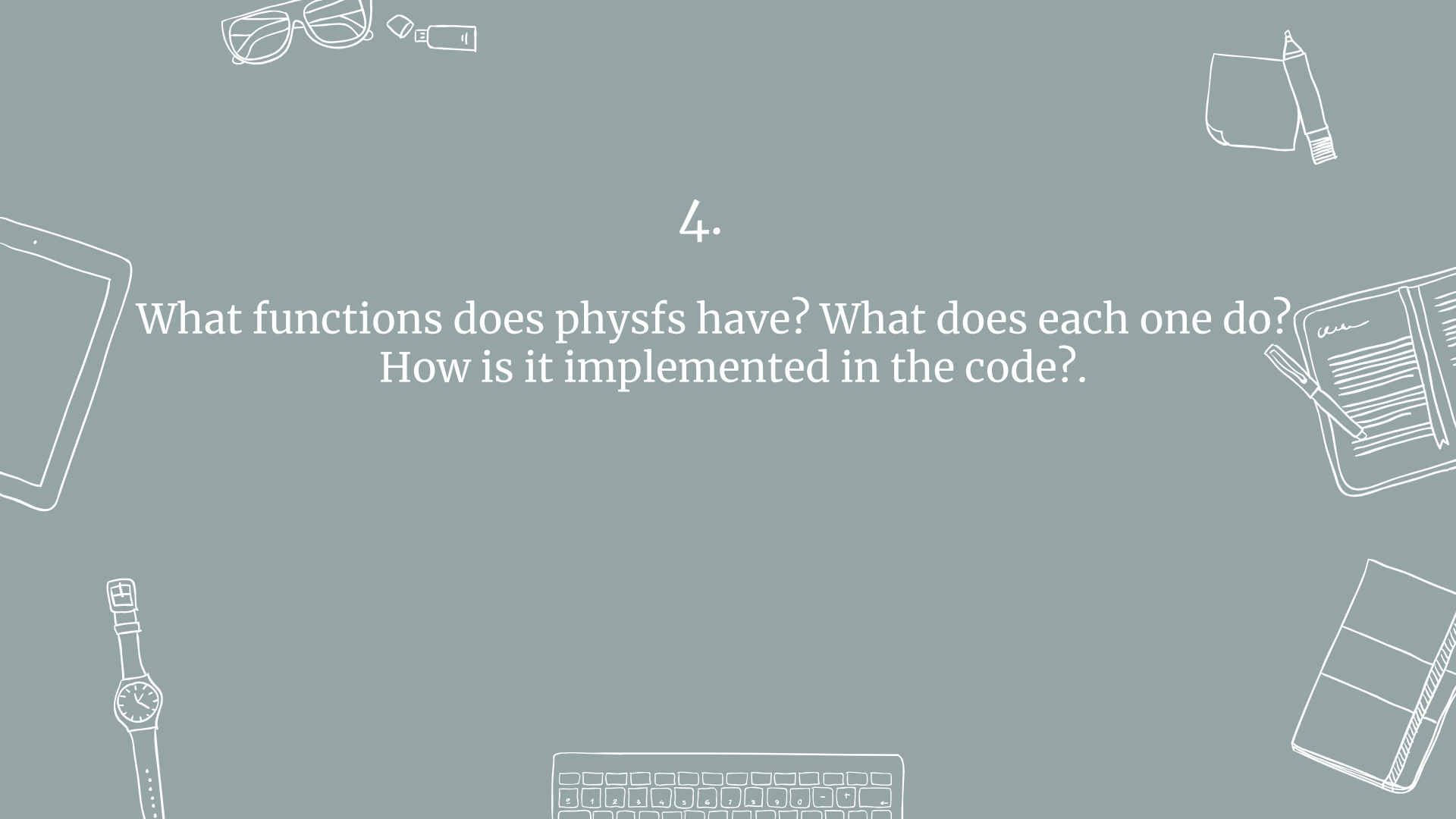
Step 5:

-Add all the .lib.



4.

What functions does physfs have? What does each one do? How is it implemented in the code?.



4.1 - FUNCTIONS OF PHYSFS

1. PHYSFS_init(const char* argv0):

This function initializes the library, and should be called at the beginning.

- **Input:** The argv[0] string passed to your program's mainline. This may be NULL on most platforms.
- **Output:** Nonzero on success, zero on error, the error can be specified with the function PHYSFS_getLastError().

```
PHYSFS_init(nullptr);
```

---

2. PHYSFS_deinit():

Deinitialize the PhysicsFS library, closes any files opened via PhysicsFS, blanks the search/write paths, frees memory.

- **input:** Don’t need any element per parameter.
- **output:** Nonzero on success, zero on error, the error can be specified with the function PHYSFS_getLastError().

```
PHYSFS_deinit();
```

4.1 - FUNCTIONS OF PHYSFS

3. PHYSFS_mount(const char * newDir,const char * mountPoint,int appendToPath):

Add an archive or directory to the search path.

- **input:**

  - NewDir: Directory or archive to add to the path.

  - MountPoint: Location in the interpolated tree that this archive will be "mounted". NULL or "" is equivalent to "/".

  - AppendToPath: Nonzero to append to search path, zero to prepend.

- **Output:** Nonzero if added to path, zero on failure (bogus archive, dir missing, etc). Use PHYSFS_getLastErrorCode() to obtain the specific error.

```
PHYSFS_mount("Assets.zip",NULL, 0);
```

4.1 - FUNCTIONS OF PHYSFS

4. PHYSFS_openRead(const char* filename):

Open a file for reading.

- **input:**
  - Filename: File to open.
- **Output:**
  - A valid PHYSFS_File * on success, NULL on error. Use PHYSFS_getLastErrorCode() to obtain the specific error.

```
PHYSFS_file* data_file = PHYSFS_openRead(fileName);
```

5. PHYSFS_close(PHYSFS_file* handle):

Close a PhysicsFS filehandle.

- **input:**
  - handle: File to close.
- **Output:**
  - Nonzero on success, zero on error. Use PHYSFS_getLastErrorCode() to obtain the specific error.

```
PHYSFS_close(data_file);
```

4.1 - FUNCTIONS OF PHYSFS

6. PHYSFS_fileLength(PHYSFS_file* handle):

Get total length of a file in bytes.

- **input:**
  - Handle: handle returned from PHYSFS_open*() - (PHYSFS_File *).
- **Output:**
  - size in bytes of the file. -1 if can't be determined.

```
int file_lenght = PHYSFS_fileLength(data_file);
```

4.1 - FUNCTIONS OF PHYSFS

7. PHYSFS_Read (PHYSFS_File* handle, void* buffer, PHYSFS_uint32 objSize, PHYSFS_uint32 objCount):

Read data from a PhysicsFS filehandle, the file must be opened for reading with PHYSFS_openRead(const char* filename).

- **input:**
  - Handle: handle returned from PHYSFS_openRead().
  - Buffer: buffer to store read data into.obj
  - Size: size in bytes of objects being read from (handle).
  - ObjCount: number of (objSize) objects to read from (handle).
- **Output:**
  - number of objects read, -1 if complete failure.

```
int readed = PHYSFS_read(data_file, *buffer, 1, (int)file_lenght);
```

4.2 - FUNCTIONS OF SDL

Before we only had to pass the file path, now we must pass the SDL_RWops structure.

Textures

```
IMG_Load(path);
```

```
IMG_Load_RW(app->Assets->Load(path),1);
```

Audio

```
Mix_LoadMUS(path);
```

```
Mix_LoadMUS_RW(app->Assets->Load(path),1);
```

In the case of the pugi library, we will have to pass the information buffer directly and delete it later.

XML

```
mapFile.load_file(path);
```

```
mapFile.load_buffer(buffer, size);
```

4.2 - FUNCTIONS OF SDL

SDL_RWops

A structure that provides an abstract interface to stream I/O. RWops might be fed by a memory buffer, or a file on disk, or a connection to a web server.

- **IMG_Load_RW**(SDL_RWops* *src*, int *freesrc*): This can load images from memory buffer.
- **Mix_LoadWAV_RW**(SDL_RWops* *src*, int *freesrc*): This can load WAVE, AIFF, RIFF, OGG, and VOC formats from memory buffer.
- **Mix_LoadMUS_RW**(SDL_RWops* *src*, int *freesrc*): This can load WAVE, MOD, MIDI, OGG, MP3, FLAC from memory buffer.
- **SDL_RWFromConstMem**(const void* *mem*, int *size*): this prepares a read-only buffer memory buffer for use with SDL_RWops.

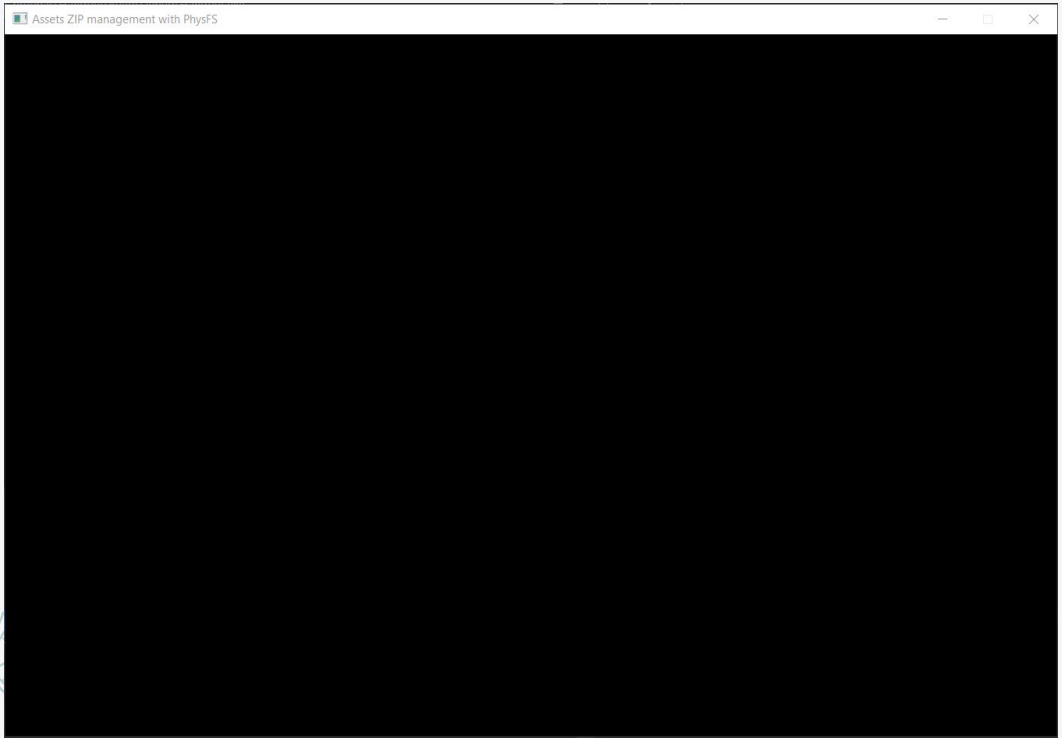
```
return SDL_RWFromConstMem(buffer, size);
```

5.

TODOS

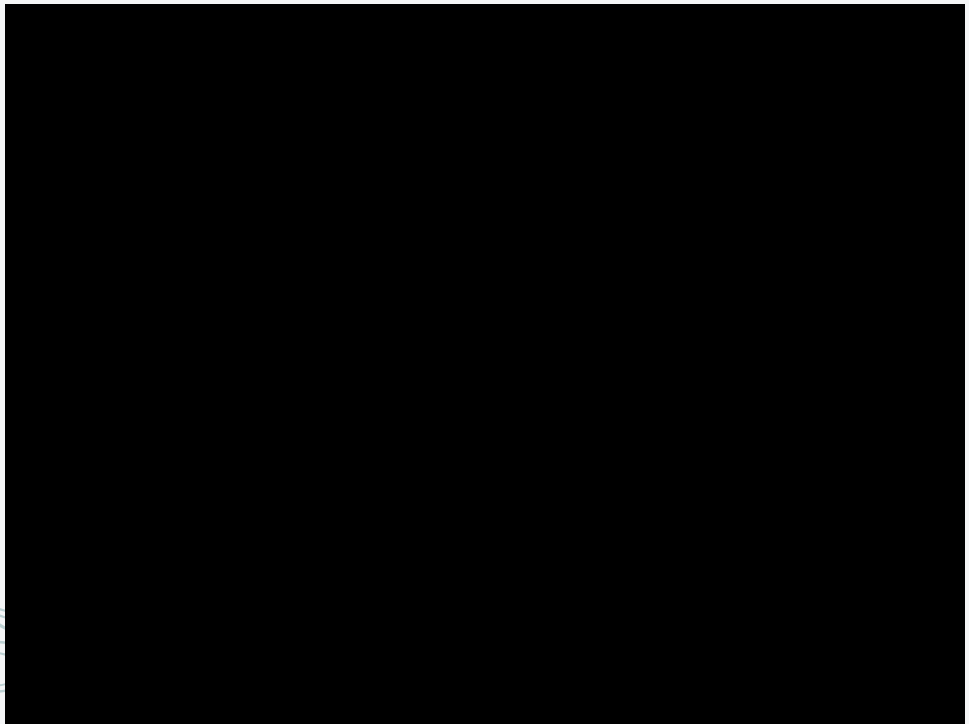
5.1 - Handout (Before TODOS)

This is what you will find when you compile the handout.



5.2 - Expected solution

This is the solution that is expected after doing the TODOS.



5.3 - TODOS

TODO 1:

Initialize the Phys Fs library.

```
AssetsManager::AssetsManager(aplication* APP) :Module(APP)
{
    //TODO 1: initialize the library (1 line)

}
```

5.3 - TODOS

TODO 2:

Deinitialize the phys Fs library.

```
AssetsManager::~AssetsManager()
{
    //TODO 2: Deinitialize the library (1 line)

}
```

5.3 - TODOS

TODO 3:

Add a path to the library.

```
bool AssetsManager::Init()
{
    //TODO 3: add the search path (1 line)

    return true;
}
```

5.3 - TODOS

TODO 4:

Call the function MakeLoad to obtain the information buffer and be able to return a SDL_RWops structure for the textures and the audio.

```
SDL_RWops* AssetsManager::Load(const char* fileName)
{
    //TODO 4: call the MakeLoad functionand get the RWops structure to load the data(3 line)

    return nullptr;
}
```

5.3 - TODOS

TODO 5:

Obtain the information buffer with the library functions and return the size of this.

```
int AssetsManager::MakeLoad(char** buffer,const char* fileName)
{
    int ret = 0;
    //TODO 5: get the information buffer, and its size with the functions and variables of the library (5 line ~)



    return ret;
}
```

5.3 - TODOS

TODO 6:

Get only the information buffer, then load the XML file, finally delete the information buffer.

```
bool Map::Load(const char* filename)
{
    bool ret = true;
    SString tmp("%s%s", folder.GetString(), filename);

    char* buffer = nullptr;
    int size=app->Assets->MakeLoad(&buffer, tmp.GetString());

    //TODO 6:load xml file through buffer (1 line)
    pugi::xml_parse_result result;
```

6.

Conclusion

6.1 - Conclusions

Phys Fs is a very interesting library to manage zip files very easily,but for the file sizes we use it may not be so essential.

Thank You