H3rtz.stdio

Book of Specifications

Jean "Areas" Bou Raad
Kevin-Brian "KB" N'Diaye
Thanh Lam Nguyen"dontneedone"
Yabsira Alemayehu MULAT "Yabs"

Contents

1 Introduction

In this book of specifications, the main aspects of our project will be presented. This will include a global presentation of the project, its objectives, a presentation of each member of the team and the distribution of the tasks among the members. At this early stage of the project, a too detailed plan will not be provided to avoid any over-promises. As this is the second version, some aspects were changed upon additional research on the subject at hand. This book of specifications will still attempt to present a clear enough idea of the project and the reason behind its transformation.

First and foremost, our team's project has one main objective, that is to be able to play .wav and .mp3 files and convert the files back and forth using an interface similar to an audio player. This project also includes a website that shows different updates and guides on how to use the program. In this book of specifications, there will be a detailed explanation on every step of the project considering different modifications.

Furthermore, this semester project will also be a great opportunity for us members to enhance our communication skills. Indeed, practicing for the presentations we will have to give in front of the jury will allow us to learn how to better communicate an idea and to learn to be as convincing as possible. On the other hand, this will be a great opportunity for the members to learn how to work in a team, since after obtaining our degrees, we will have to work in corporations with numerous people on different projects.

In conclusion, all the necessary information required to understand the flow of our subject is provided in this book of specification with precise and detailed explanations.

2 Outline of the Project

2.1 Context and definition of the project

We are four students in S4 at EPITA who want to present our project to you. We thought that handling audio would be a great bridge between what is needed and what we have learned so far.

2.1.1 Members of the group

Kevin-Brian N'DIAYE (Group Leader)

I started learning about computer science at 14 with basic projects such as Unity or web development. However, these didn't pick my interest until I learned about machine learning and robotics in 10^{th} grade and its limitless potential. Audio seemed like an small yet, deep topic I feel ready to get into. Jean's excitement about the project easily got to me and I can't wait to see how it goes.

Favorite quote : I never loose, I either win or learn.

Thanh Lam NGUYEN

How did I end up in computer science ? Easy! My brother was playing with his intelligent robot while I was wondering if 12 years+ of studies to be a surgeon is what I really wanted. There, I started doing some researches, and the logic and the desire of something new got me here in EPITA. For this semester the first thing I wanted is to be in a motivated group. Done.

Now we chose our subject, I am excited to learn something new ! About something I never thought I will work on signals, audio player. I feel good with our group project and I am confident to say that this project will be a good experience too!

Favorite quote : "Just take a seat, I got this." Phoenix

Yabsira Alemayehu MULAT

After 1 year and half experience of learning computer science, there is nothing exciting like working in a group project. Learning computer science was not my first choice but once I joined, I started appreciating the subject a lot. This kind of projects makes it even more exciting because it gives me the opportunity to work with different amazing coworkers and learn many new things. Out of all the languages that I learned so far C is kind of difficult for me because it is not something that I had experienced before. But I am sure that this project will make me work hard and learn the language to the fullest. Furthermore, this project requires different algorithms to learn and apply. This will be really important for me because the more I learn different algorithms the more I become effective and fast in programming as well as a good substance for my resume. In the process of making this project successful, I will be able to make a lot of improvements on myself and with people (my teammates) which is really important for my future jobs. It's like hitting two birds with one stone. All in all, I am really excited to work on this project and learn a lot of things. I am certain that it will be an amazing journey for me and my teammates as well.

Favourite quote : "That which does not kill us makes us stronger."

Jean Bou Raad

Computer science and problem-solving have always been a passion for me. Before entering EPITA, I learned a bit about programming languages such as C#, node-js, etc... I have put that knowledge to use during the two lasts projects (S2, OCR), and I will do that once again for this one. I have learned a lot about C during OCR and would love to deepen my knowledge of this language.

I am responsible for the idea of this project because of my passion for music. I wanted to bring it to computer science, and an audio player seems like a perfect way to do it. The other team members accepted it despite having diverse ideas, so I am grateful for that. I will be mainly working on audio codecs and will bring some help to any team members if needed.

Favourite quote : Never gonna give you up. Never gonna let you down.

2.2 Objectives

The final project should be able to play .wav and .mp3 files and convert the files back and forth. The GUI will be very similar to a normal audio player with options like :

1. Fast-forward
2. Go back a set amount of seconds
3. Adjust speed

This will require encoding/decoding .wav and .mp3 files and advanced compression in order to go from .wav to .mp3. Handling .wav files will be our main priority to understand how everything fits together.

2.2.1 Limits

1. Only two codecs

2.2.2 Needs

1. GTK / Glade : Default option for GUI in C
2. Pulseaudio : Default library (available on linux) to play the raw data (final result of decoding)
3. Gstreamer: default media library in C, used for universal audio decoding to wav.
4. Documentation on each file format and audio compression for encoding/decoding
5. Bootstrap: web framework to build our website
6. Okteta: application to read raw data
7. Github Pages: website hosting

2.3 State of the Art

In the last two decades, we have been able to achieve lossless audio compression using Psychoacoustics and MDCT (Modified discrete cosine transform). Based on those two concepts, audio formats such as FLAC or AAC have emerged.

- 1926 : Paul M. Rainey patens a machine that uses 5-bit PCM (Pulse-code modulation). It's the first method to represent analog signals digitally.
- 1950 : C. Chapin Cutler at Bell Labs creates the DPCM (Differential pulse-code) that efficiently handles digital and analogue input.
- 1974 : Nasir Ahmed, T. Natarajan, K. R. Rao develop the DCT (Discrete cosine transform).
- The 1970s and 1980s : Perceptual coding (or psychoacoustics) raises the compression ratio using algorithms such as APC (Adaptive Predictive Coding) and CELP (Code-Exited Linear Prediction) used in MP3 and AAC.
- 1986 : J. P. Princen, A. W. Johnson and A. B. Bradley improve the DCT into the MDCT (Modified discrete cosine transform), method now used in most modern file formats.

On Linux, the best media players can be sorted by:

1. Number of files formats supported
2. Degree of customization
3. Audio filters
4. Subtitle synchronization

Based on those criteria, the best media player might be VLC Media Player:

- supports every audio format
- cross-platform
- highly customizable
- Advanced control

2.4 Parts of the project

It boils down to four major aspects : The audio decoding, audio encoding, the GUI, and the website. Precisely, we have :

- WAV codec support (encoding & decoding)
- MP3 codec support (encoding by hand, decoding by Gstreamer)
- The User Interface
- Asynchronous player
- Conversion support
- The Website

Task management is key to accomplish such a project especially in the beginning. Therefore, breaking down tasks into smaller ones is essential to stay organized and to keep a steady pace.

1. **WAV codec support :**
   Encoding and decoding WAV files and play them.
   This codec will probably be the easiest one to implement. It is a container format, meaning that the data inside the file is uncompressed so this part will not require too much algorithms. Instead, It will allow us to approach the common concepts about audio codecs and how to play audio correctly. For encoding, it will be mostly about understanding headers and using I/O correctly.

2. **MP3 codec support :**
   Encoding and decoding MP3 files and play them.
   MP3 (a.k.a. MPEG-1/2 Audio Layer III) is the most popular codec on the market. This format is compressed, meaning that each frame (audio sample) must be read, decompressed and then played properly. Hence, It will be harder to implement. MP3 is a standard codec with its specifications but they are vague concerning the encoding method. It requires some heavy pre-processing on the raw data with a polyphase filterbank (Fourier's transform) in addition to the classical Huffman coding used to compress data. The complete process is detailled in a later part.

3. **User Interface :**
   The Graphical User Interface must be easy to use and convenient, providing a way to input text and and display the the results of the operations entered by the user.

4. **Asynchronous player :**
   Use multi-threading for smoother user experience.
   At first glance, It seems that to make the interface work while decoding some audio requires to split processes. Multi-threading is a new notion for all group members and will require some time to understand and apply the notions around.

5. **Conversion support :**
   Going from one file format to another. WAV to MP3.
   We are aiming for an implementation from raw PCM signal to MP3 conversion. It involves many algorithms & the process is complex at least with the lack of experience we have in this domain.

6. **Website :**
   The website should be able to be a platform for our program. It will also fulfill its basic functions :

   (a) Introduce the members of the group

   (b) Explain what we are trying to accomplish with this project

   (c) Make our project available directly on the website for quick use

   (d) Have a downloadable version of it

2.5 Audio decoding process

Audio codecs can be compressed or uncompressed. The last one is the easiest to parse and play. For this kind of file (e.g. : WAV), the data is held in a file after a header.

A header provides all the information about a file. It can be an image, a video, or music: one can find a header in all those files. For audio codecs, they contain essential information such a bit sampling, sampling rate, etc...

Parsing those can be done by hand using Read function from the C99 standard. Otherwise, the problem can be tackled using the Mmap function. It will probably be the solution chosen: most formats provide a standard header format easy to copy-paste in our code using structures.

For all kinds of codec, after reading the header, the program can start reading the data. The difference between compressed and uncompressed is the processing carried over to the audio playing loop.

In most cases, compressed formats use a header per sample enabling us to retrieve useful data to decompress each sample properly. These codecs can adopt a variable bitrate (quantity of data over time. It can indicate the quality of the audio compressed for lossy formats: higher is better). Therefore, reading those formats can get hard.

2.5.1 MP3 decoding

MP3 decoding relies on several decompression techniques to retrieve lost data from the compression. Therefore, it needs a lot of algorithms.

2.6 Audio encoding process

As stated previously, uncompressed formats are easier to encode. The audio encoding requires a raw data source (**P**ulse **C**ode **M**odulation **D**ata for instance).

The process ends quickly for uncompressed formats: the program must write a header according to the format's specifications, and then it can write the rest of the data to the file. For compressed formats, the process is a lot harder. First, the global header must be produced by the program. Then, the software must start writing some data.

This data cannot be left as uncompressed, so some processing is required, which requires various algorithms. For instance, MP3 uses Fourier's transform for preprocessing and compresses the data using the Huffman coding method (binary trees). Also, each audio frame requires the production of a header. To finalize our previous example, the header is going to hold the Huffman coding table for the frame required to decompress the data.

The complete process is illustrated below:

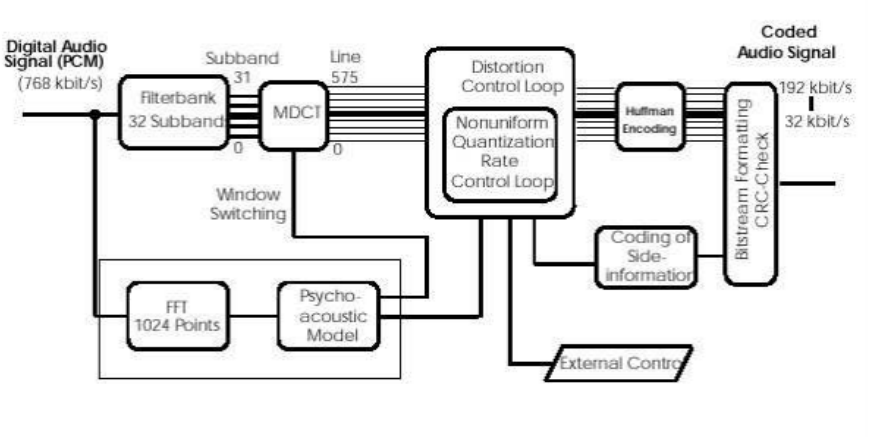

Figure 1.2: Detailed Encoder Diagram

Source: `http://www.mp3-tech.org/programmer/docs/jacaba_main.pdf`

We will try to implement the left part for the second defense and the right part for the last defense. However, most parts of the process are new to us, and we lack of a proper point of view to evaluate difficulty.
The notions tackled here involve some heavy mathematics in addition to the algorithmic difficulties. And the standard provides little to no information on the how.

2.7 Audio playing

Playing the data is a key part of our project. The solution adopted by the team is to use PulseAudio. It is a well-known and popular software available on most Linux machines (including the Archlinux distribution on EPITA's computers).

It also provides an API coded in C to pass data via an audio server. It uses classical IO methods to do so with reading and write. Its documentation can be found online at:

https://freedesktop.org/software/pulseaudio/doxygen/index.html

Using such a library was mandatory because passing audio to a driver is complicated, especially using low-level system calls in C. Currently, the team does not know yet if we will need to use the API's asynchronous version for our project. However, we already know that it will provide the required features to build this project.

2.8 Assignment tabular

We will assign a letter to every member to make our schedule more readable

1. KB : Kevin-Brian
2. J : Jean
3. L : Lam
4. Y : Yabs

Tasks	KB	J	L	Y
Multi-threading	×	×	×	×
Audio formats	×	×		
GUI			×	×
Website		×	+	

Cross : (×) Person in charge and Plus : (+) Assistant.
As you can see multi-threading will involve everybody.

2.9 Progression

Now, we need to consider how we are going to handle our time based on the three main deadlines :

1. The 1st presentation (April 5th - 9th April, 2021)
2. The 2nd presentation (3rd - 7th May, 2021)
3. The 3rd presentation (14th - 18th June, 2021)

We made a tabular to make it easy to read. Each letter represents a presentation. The goal is obviously to hit all marks for each task.

Tasks	1st	2nd	3rd
Multi-threading	30	70	100
Audio decode - WAV	80	100	100
Audio encode - WAV	30	100	100
Audio decode - MP3 (GST)	75	100	100
Audio encode - MP3	0	50	100
User Interface	40	70	100
Conversion Support	0	40	100
Audio decode - MP3 (Algo)	0	0	0
Website	100	100	100

Notes:
In red: modified tasks.
Audio decode - MP3 (GST): for the moment, we limit our project to audio encoding which is already a heavy task.

3 Conclusion

The project is about implementing an audio player which has a very interesting algorithmic side when it comes to signals. Research later taught us that encoding and decoding mp3 files would be an entire project of its own. Therefore, most of the project will remain the same but in order to guarantee a finished product we decided to remove the MP3 decoding from the algorithmic part and delegate it to a new library : GStreamer.

To sum it up, the 4 major aspects of the project are the audio decoding, audio encoding, the GUI, and the website.

So the final project should be able to play .wav and .mp3 files and convert the files back and forth. The main goal stays the same as the player will still be able to play mp3 files without us decoding it on our own. And it also needs to provide a working interface similar to an audio player.