UNIVERSIDAD AUTONOMA DE MADRID

ESCUELA POLITECNICA SUPERIOR

Grado en Ingeniería de Tecnologías y Servicios de Telecomunicación

TRABAJO FIN DE GRADO

DESARROLLO DE UN SOFTWARE DE SINCRONIZACIÓN AUTOMÁTICA ENTRE TONALIDADES Y COLORES EN MUSICOTERAPIA AUDIOVISUAL

Natalia Delgado Galán

Tutor: Mathias Funk

Ponente: Daniel Ramos

Junio 2016

DEVELOPMENT OF SOFTWARE FOR AUTOMATIC SINCHRONIZATION BETWEEN TONALITIES AND COLOURS IN AUDIOVISUAL MUSIC THERAPY

AUTHOR: Natalia Delgado Galán

TUTOR: Mathias Funk

Industrial Design Department

Technical University of Eindhoven

June 2016

Escuela Politécnica Superior

Universidad Autónoma de Madrid

Abstract

This end of bachelor project consists in the automation of music and colour synchronization designed to be used in music therapy. The idea behind this concept is using colour as a new dimension to visually interpret the complex variations in music, and this project contributes to it by improving its efficiency through automation.

Studies have demonstrated that the tensions in a musical piece are related to the emotion or the mood the piece transmits [1], the same way it has been proved that different colours induce a certain mood or emotion in people [2]. Those conclusions have been used by a doctorate student in the Technical University of Eindhoven to back up the proposal of making emotion the common variable between music and colour [3][4] for further music therapy purposes. This project is the technological part of the later research work mentioned. To develop it, a thorough understanding of the doctorate student's work is required, to then to propose a software that can fulfil its demands.

To start with, this paper studies Music Information Retrieval research, MIDI files format and the relation between them. This is necessary to develop a program capable of reading a MIDI file and converting it into a list of notes with their corresponding timings. Afterwards, high level music features such as chord and tonality changes are extracted, making use of music theory knowledge to reinforce MIR methods. Once the MIDI file has been interpreted into a list of chords, they are expressed as musical intervals, i.e. relative distances between them. This step is done prior to carrying out an automatic mapping of musical intervals to colours, following findings and conclusions to make a coherent matching between both disciplines [3][4]. The music to colour mapping results as a list of colours associated to timings. Finally, to visualize the outcome of the previous steps, coloured lights are changed following the colours list while the music is synchronously played. This last section is controlled by a software that establishes a connection with the lamps via Wi-Fi and executes the change-colour commands.

To finish with the project, an evaluation of an external software employed is carried out using a method based on speaker diarization. Finally, conclusions regarding the expectations of the project are made, and ideas for future work and improvement are suggested.

Resumen

Este Trabajo Fin de Grado consiste en la automatización de la sincronización entre música y color, diseñada para fines relacionados con la musicoterapia. La idea detrás de este concepto es utilizar el color como una nueva dimensión capaz de interpretar visualmente las complejas variaciones en la música, y este proyecto contribuye a ello mejorando su eficiencia a través de la automatización.

Estudios han demostrado que las tensiones que aparecen en una pieza musical están relacionadas con una emoción o estado de ánimo [1], del mismo modo que se ha demostrado que determinados colores generan una emoción o un estado en la gente [2]. Estas conclusiones han sido utilizadas por una estudiante de doctorado de la Universidad Técnica de Eindhoven para apoyar su propuesta de utilizar la emoción como la variable común entre música y color [3][4] para contribuir a los avances de la musicoterapia. Este proyecto es la parte tecnológica del trabajo de investigación que acaba de ser mencionado. Para desarrollarlo, un minucioso entendimiento de dicho trabajo es necesario, para después poder proponer un software que se ajuste sus necesidades.

Para comenzar, en esta memoria se estudian las bases de *'Music Information Retrieval'*, del formato MIDI, y de la relación entre ambos campos. Este estudio es necesario para desarrollar un programa capaz de leer un archivo MIDI, y convertirlo a una lista de notas con sus correspondientes marcas en el tiempo. Después, características musicales de alto nivel, como son detección de acordes o de tonalidad, son extraídas con ayuda de conocimientos de teoría musical para reforzar los métodos de MIR. Una vez el archivo MIDI ha sido interpretado como una lista de acordes, se expresan en forma de intervalos musicales, es decir, distancias relativas entre ellos. Este paso se realiza justo antes de llevar a cabo un mapeo entre intervalos musicales y colores, siguiendo los resultados concluidos por investigación [3][4] para poder emparejar ambas disciplinas de forma coherente. El resultado de dicho mapeo es una lista de colores asociada a sus correspondientes marcas en el tiempo. Finalmente, para visualizar el resultado de los pasos anteriores, luces de colores cambian siguiendo la lista de colores mientras la música suena de forma sincronizada. Esta última sección es controlada por un software que establece una conexión con las luces vía Wi-Fi y que ejecuta los comandos cambio-de-color.

Para concluir con el proyecto, se realiza una evaluación del software externo utilizado, empleando un método basado en '*speaker diarization'*. Por último, se desarrollan conclusiones respecto a las expectativas del proyecto, y se proponen ideas y mejoras para un trabajo futuro.

Keywords

MIDI event, delta timing, music information retrieval, automation, synchronization, High-level music content descriptor, low-level audio feature, beat tracking, chord detection, musical interval, tension-release relation , music to colour mapping, diarization.

Palabras clave

MIDI event, delta timing, Music Information Retrieval, automatización, sincronización, descriptor de contenido musical de alto nivel, caracterísitcas de audio de bajo nivel, seguimiento de tempo, detección de acordes, intervalo musical, relación tensión-liberación, mapeo de música a colores, diarización.

Acknowledgments

I would like to thank the Tue team: Katarina, for working and dreaming with me as a colleague and as a friend; Mathias for helping, teaching first notions of music technology, and for your patience with this Holland-Spain tutoring work; finally, Eggens for having faith and offering the opportunity for this to happen.

Thank you Daniel Ramos, for giving the freedom to do something new and different.

To all the people with whom I lived the Eindhoven experience. Who would ask with interest and would care about my work. For those who shared the excitement when I was told I had the chance to do this. Thank you for keeping my energy bright.

To all my friends in Spain who have always been there and whom I love. You have the power flip my mood in the most positive way.

To the musical world, for letting me be part of its beautiful wonders.

And last but not least, thank you family. For your support during this project and always. For your most sincere advice, for giving the greatest hugs, and for bearing my worst tantrums.

Every small thing counts. Thank you.

CONTENT INDEX

FIGURES INDEX

TABLES INDEX

EQUATIONS INDEX

1 Introduction

1.1 Motivation

In search of an improvement in music therapy material, doctorate student Katarina Biljman [3][4] has been investigating in the Technical University of Eindhoven the possibility of increasing musical experience by means of a new dimension: colour. There is faith that someday, this concept will actually be brought to music therapy offices in the form of a whole, compact product, based on finding a parallelism between the emotion music and colour suggest.

1.1.1 Music therapy and colour

Before going any further, 'why this product?' 'Why contribute to music therapy?' 'Does it actually work and does it have any fundamentals?' 'Why do we know or believe this program will be useful?'

As stated in the book 'A comprehensive guide to music therapy' [1] *"research in the broad field of music therapy has been a real focus of attention over the last forty years, and a number of clinicians and academics have obtained whatever resources they could find to initiate, carry out and report studies."* The number of documented studies reported in databases, journals and books is significantly increasing, reflecting the growing interest of both professionals and patients, and giving a hint of the discipline´s validity. Furthermore, while this discipline grows and multiple methods and directions appear, research on proof of its effectiveness is also augmenting with positive results. All of this leads to believing in the benefits of music therapy and feeling a motivation to contribute to a valuable aspect such as working on the improvement of other individual's health.

Furthermore, if music affects the brain and can influence on mood and emotions, does this mean colour can take this effect of music to the next level? Despite the little research developed on the expression of music through colour, in the recent history, a number of individuals have showed their urge to express the relation between colour and music, such as the painter Kandinsky or people that suffer from synaesthesia. Even Spielberg reproduced a conversation between humans and aliens as a duet of music and visuals in his film Close encounters of the third kind [5]. The reiterative appearance of people trying to combine music and colour indicates that the relation between them is probably not random. As a matter of fact, a new art form, called Lumia, that that supports this thought has risen, which permits visual improvisation imitating music. Fred Collopy compiles in his article *Color, Form, and Motion* [6] much research on this new art form explaining the history and proof behind it. Even though, as Collopy states, there is little consistency among the different propositions of mapping colours to music in the past, there is a strong belief that this mapping has a real basis behind, despite not being an established and consolidated one. This awakens motivation in moving in that direction and exploiting its capacities by coordinating music and colour to generate a more powerful healing art.

An interesting approach on colours to music mapping was proposed by Brian Evans [6], based on the frequently occurring idea in the temporal arts of tension-release. This idea was then used to produce a colour grammar for further coordination with other researchers. The same way Brian Evans proposes a colour grammar based on tension-release, Katarina Biljman states in her work that *"tension and resolution is considered as one of the core*

principles for evoking emotions by music" [7]. Connecting the similarities between those two concepts, she proposes her own 'colour grammar': *"Due to the contrasting relation of the two functions,"* (tension and resolution) *"they were translated into two complementary colours of the spectrum"*. While her project aims at establishing the best relation possible based on solid foundations for future contributions in music therapy for autistic children, this project could be expressed as the technological approach of her work. This is, a study and comprehension of the previous, followed by an automation and synchronization of colour and music.

1.1.2 Why automate and synchronize?

Stepping aside of investigation and valuable conclusions, and moving further on to the application of these, issues appear. Whether it is used for therapeutic purposes or for entertainment concerts, the project lacks efficiency and usability.

It would seem reasonable that an expert musician would be able to put into practice these methods successfully once he or she had comprehended the conclusions and theory proposed. However, the outcome of such effort would be far from economic, resulting inaccessible for most patients or music therapists. Hence, a technological approach is missing in the fulfilment of the product. In order to bring this concept into an actual software that a day to day person could afford to use or purchase, two elements are missing in the system: efficiency and usability. To solve the problem of efficiency, automation is proposed, being the first objective of this project. It must be also taken into account that the outcome of this project is aimed to be used by non-engineers, and therefore it should result as a user friendly interface for anyone to understand and use .This concept is addressed to as usability and it is the second general objective of this project.

Another motivation for the later objectives is explained in this paragraph. The project was requested to be tried in an orchestra concert where coloured lights would ambience the room to enhance the musical experience. The problem was that lights were changed manually following a chart of timings and colours. The event ran smoothly but it required a person with consistent full attention and precision. Therefore, developing a software that executes those commands automatically and can be easily used by any person is also proposed to cover the efficiency and usability matters mentioned before applied to this case.

1.2 Objectives

From the previous paragraph it is deduced that the general objectives of this project are providing previous work with efficiency and usability to transform concepts and conclusions into a product that executes them.

At a smaller scale, and breaking down the general goals into steps, the following objectives are proposed:

- Automatic extraction of harmony information (chords and change in tonality) from a classical musical piece contained in a midi file.

- Automatic mapping of music tension-release patterns to colour
- Automatic synchronization of coloured light changes with music harmony changes
- Evaluation of software employed performance

1.3 Structure

This project is part of a bigger project. As explained on the objectives section, this project conforms the automation/engineering section of the structure of the bigger project and it is composed of the following:

1.3.1 Objectives and motivation

Why should this project be developed? What moves this idea? What is behind this concept? These matters are dealt here.

1.3.2 State of the Art

Where are we starting from? In what knowledge are we based on in order to start investigating? How can we prove our conclusions are based on solid facts? This section displays the research carried out in order to perform the following work.

1.3.3 Design and Development

This section contains the explanation of the development and design of the project. It describes the procedures and programs used and carried out, as well as the motivation for choosing them over other alternatives. To explain everything as clear as possible, a brief description of the starting point and the target are included at the beginning of each step of the development.

The design and development section is divided in three blocks: the first consists in the automatic chord extraction of a musical piece; the second carries out a music-to-colour mapping; and the third executes the previous mapping to make coloured lights change synchronously with the music. These blocks are explained below:

- Block 1: The objective of this section or block is to carry out a musical analysis of an audio contained in a MIDI file. To do this, the block is divided in three parts. The first part, which is developed in Processing, reads the MIDI file and extracts the notes contained in it, together with its corresponding timings. This allows exporting a file containing the relevant information of the MIDI file that Matlab is able to read. The second part of this block is developed in Matlab, where chords per bar are extracted from the notes the MIDI file contained. Finally, the third part of

this block explains the method used by an external software used in this project to solve the problems encountered in the previous part.

- Block 2: This section or block is developed in Matlab. Here, a mapping of musical tension to colour tension is made. The design is based on findings and conclusions [7], and a proposal to automate them. The development shown follows that design.

- Block 3: This section or block is also developed in Processing. This program is designed to send signals to coloured light via Wi-Fi with the order to change colour.

A diagram illustrating this structure is represented in the Design and Development section.

1.3.4 Tests and results

A software implemented to test the quality of external tonality detectors used is presented in this section. First, the method employed to evaluate the software is explained. This method is referred to as chord diarization and it is based on speaker diarization. Afterwards, a table with results included to contrast with musical expert's results.

Also, a demo video was made as a result of this project. An explanation of how it was made and what relevant features it contains is also included in this chapter.

1.3.5 Conclusions and future work

What can we conclude from this work? What was carried out as planned? Was the load of work misjudged? Were the objectives achieved? The evaluation and answers to these questions are presented in the fifth chapter.

What could be improved and how? In which direction would future work be focused? Suggestions for improvements in the future are also made in this last chapter.

1.4 Summary

The motivation of this project could be resumed as a combination of humanistic and technological concerns: the desire to contribute to the world of music therapy in an innovative way, by employing automation and synchronization to make the idea possible. These motivations lead to four main objectives: automation of harmony information, mapping of music to colour, synchronization of coloured light to music, and evaluation of external software employed.

As well as establishing the objectives for this work, a study on the matters used is made before starting designing or developing the project. It is important to know in detail the state-of-the-art of the field being investigated in order to contribute to it and this is the reason for including the next chapter in this paper.

2 State of the Art

In order to achieve the objectives proposed for this project, a previous study is required to comprehend where earlier research has reached to, what knowledge could be useful and finally be able to conclude which would be the best starting point for this work.

It was decided that MIDI is the most appropriate format for this project due its high compression capabilities, its frequency 'cleanness' and its promising increasing use in the future. These advantages of MIDI will be further explained in the following section. It is convenient to understand what MIDI is, what its possible uses are and how it is structured. Hence, a review on MIDI theory is included below.

Moreover, the investigation line concerned with the extraction of musical features such as melody or chord extraction, Music Information Retrieval is also of interest for this work. To generate and understand software that is able to extract features of this type, whether it is to a simpler or more complex level, it is convenient to be aware of the most recently used techniques in this field. Therefore, theory on various aspects of MIR are also included in the section below.

Finally, the relation between MIDI and MIR, and what can MIDI provide with to MIR will be explained in the part of this section.

2.1 MIDI: uses, format and advantages

2.1.1 What is MIDI?

MIDI stands for "Musical Instrument Digital Interface". As the name indicates, it was created for the sake of musical technologies. Electronic music began with the start of technology in the beginning of the 20th Century, and it wasn't until 1983 that the International MIDI Association created MIDI for better coordination amongst people in the music-technology world.

It is convenient to highlight that even though it is a common mistake to confuse it with a musical language, we are actually talking about a very powerful communications protocol, but not a language. It could be defined as *"an agreement among manufacturers of music equipment, computers and software that describes a means for music systems and related equipment to exchange information and control signals"* [8]. In other words, an established 'code' for music people to understand each other.

Joseph Rothstein offers a very interesting metaphor between audio information and MIDI data in his book *MIDI a comprehensive introduction* [8]. He expresses that while the recording of a pianist playing is a capture of the musical information itself, the MIDI data could be compared to the music sheet the pianist is playing from. What a music sheet stores *"is not the sound of the piano music, but the instructions necessary for a human performer to re-create that music"*. MIDI data could be considered as an 'extended version' of this concept. More than a description of musical sounds, it is a set of instructions to generate sounds in electronic steps. I.e. having a 'virtual professional interpreter' and a MIDI file, the product of these could be compared to the outcome of the

recording of a pianist. The virtual professional interpreter is known as a bank of sounds or a sample, which depending on its quality will reproduce more or less accurately the playing of a real pianist. Below is an illustration of this concept.

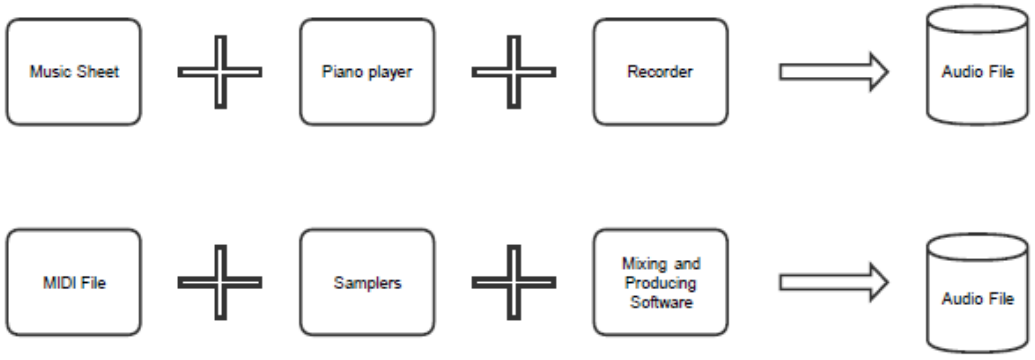

Figure 1: A comparison of music sheet and MIDI files

2.1.2 How does MIDI work? MIDI format

A MIDI file consists of a list of events that occur along a time scale measured in ticks. These events mainly inform of each note recorded in the file, where each event contains the data of whether the note starts (note on) or it ends (note off), of its value (frequency) and its moment in time (timestamp measured in ticks, the unit in a midi file).

Thoroughly understanding the standard file format of midi is essential both to interpret and to work with MIDI files. Hence, below is an explanation of the format of midi files.

1. Chunk Types

A standard MIDI file is divided into series of 8 bit-bytes called chunks. These chunks can be either header chunks (MThd) or track chunks (MTrk). Each chunk is formed by a 4 character ASCII type and a 32bit length. In the following section both chunk types will be explained with the information contained in them.

a) Header Chunks (MThd): The information in the header chunk is referred to the entire MIDI File. It contains the number of tracks in the file, the length of the header, and the division type. In the table below, main information contained in the header chunk is presented.

Table 1 : MIDI header chunk format

Bytes	**Syntax**	**Description**
1-4	Chunk Type	Always ‘MThd’
5-8	Header Length	It is a 32-bit representation of the number of bytes of data that follow in the header chunk (type and length excluded).
9-10	Format	Used to organize multi-track music, three formats are used (* formats explained below)

11-12	Number of tracks	Gives the number of track chunks contained in the file.
13-14	Division	Gives the meaning of the delta time (**number of ticks per quarter in case of format 0 and PPQ, or number of ticks per frame in case of format 1 and SMPTE).

Below are explained some concepts from the table above.

* Format clarifications

- Format 0: considered vertically one dimension. Mostly used in monophonic music. In the case of multiple voices, they should all be condensed into one track, hence not being the best format.

- Format 1: intended for multiple voices, essential for music with melodically independent tracks, where each voice occupies one track

- Format 2: considered horizontally one dimensional. Used in music with temporally independent tracks.

** Division clarifications

In the case of format 0, division specifies an ideal metrical time (expressed as the number of ticks per quarter, PPQ)

In the case of Format 1, division specifies a time code based measured in SMPTE (expressed as the number of ticks per frame). This number is always negative.

The concepts delta time, PPQ and SMPTE will be explained in the following section.

Below is an example of a header chunk. As it can be observed, midi files are written in binary code and expressed in hexadecimal.

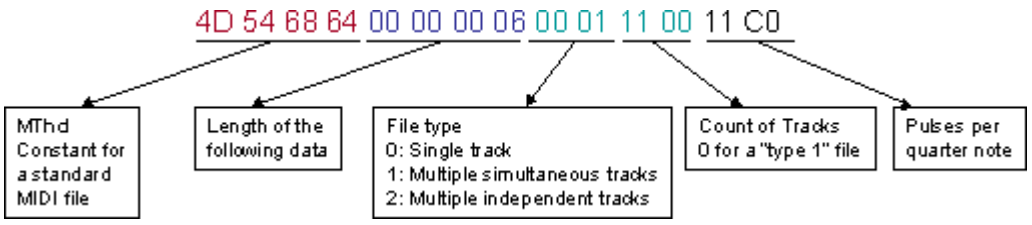

Figure 2: Example of header chunk in binary code with explanation of each part [10]

b) Track Chunks (MTrk): The sequential data is stored in track chunks. This is information of note on and note off commands, and other less frequent events that might occur and be stored in the file along the musical piece. Every track chunk is composed by a chunk type, a track length, and a list of MTrk events. In the table below, main information contained in the header chunk is presented.

Table 2: MIDI track chunk format

Bytes	Syntax	Description
1-4	Chunk Type	always 'MTrk'
5-8	Track Length	It is a 32-bit length representation of the number of bytes of the track.
9-end	MTrk Events	Multiple events, all with the same structure: Delta time + Event.

Below is illustrated an example of the beginning of a track chunk, this is common to all track chunks.

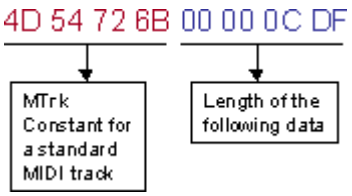

Figure 3: Example of the beginning of a Track event [10]

*Clarification for MTrk Events

[Bytes 9 onwards] MTrk Events: they always have the same structure <Delta Time> <Event><Delta Time> <Event><Delta Time> <Event> … There are many MTrk Events for a sole track chunk. In some files there can be just one track chunk, and all the events of the file are contained in it.

Below is illustrated an example of the second part of a track chunk, containing time information and an event. In this example separate events are shown, however, in a real case many delta time and events go together one after the other.

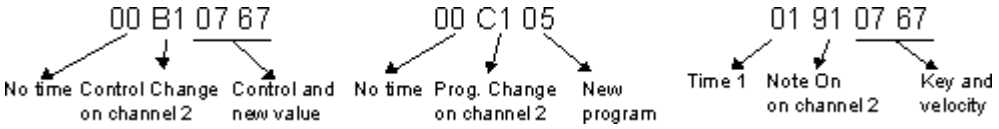

Figure 4: Example of three possible options for the second part of a track chunk [10]

Even though there are a number of types of events that will be explained in the next section, the kind of event we will mostly come across with is a Note On or Note Off event. The last of these three examples of midi events is the one we are interested in.

2. Event Types

In the track chunks, as seen above, the MTrk events are always composed by a delta time and an event. Delta time is common for all kinds of events, however, there are three types of events: (a) MIDI events, (b) meta events, (c) system-exclusive events.

a. Midi Events: These events give information about each note played. They must contain five items of information: 1) Delta time, 2) Note on/note off status, 3) Channel/track number, 4) Note number (pitch), 5) Attack velocity (dynamics)

1) Delta time: "indicates the time delay between the onsets of discrete MIDI events in a serial stream of data "[9]. This means delta time is a parameter that gives a timestamp to all events in a midi file and can be interpreted into notes in time by use of the value PPQ or SMPTE. As explained above, depending on the format of the file (information explicit in its header) the delta time will be expressed in pulses per quarter note (PPQ) or in SMPTE time code.

If the value format is 0, the delta time will be expressed in PPQ. I.e.: if delta time is 20 and in the header the division value is 8000 microseconds, the real time between the previous event and the executing the current event from is 8000*20 microseconds.

If the value format is 1, the delta time will be expressed in STMPE. I.e. the delta time expresses the same as in the case of PPQ with the extra condition that the delta time value is multiple of the frame rate times the sub-frame rate. This means that the number of ticks (division value extracted from header) and number of quarter notes (value of delta time) are still needed.

In conclusion, the time at which a MIDI event occurs depends on three things: the number of ticks that the event itself carries; the time of the previous event; and the time division of the whole MIDI song.

Note that the MIDI delta time is "delta" time and not "absolute" time. It specifies the time after the previous event and not the time from the beginning of the MIDI sequence. Hence, previous data is needed to define the time of an event.

2) Note on/note off status: This value indicates whether the note is being pressed or released in the moment the delta time indicates.

3) Channel/track number: There are up to 16 possible channels or tracks in midi files. This value indicates to which channel or track the note pertains to.

4) Note number (pitch): The pitch of musical notes is represented with a numerical value from 0 to 127. The figure below illustrates this concept.

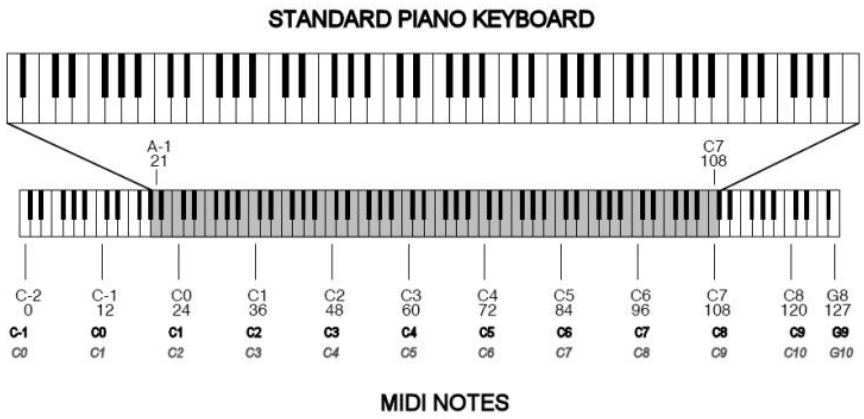

Figure 5 : Standard Piano keyboard notes to MIDI notes

5) Attack velocity (dynamics): this value indicates the intensity or loudness of the piece. It ranges from 1 to 127, being 1 *pianissimo* and 127 *fortissimo* in musical terms.

b. Meta Events

These events include in the file extra information such as track names, tempo indications, time-stamped lyrics, copyright notices, and so on.

c. System-exclusive Events

These events include hardware addresses that are often specific to one manufacturer.

Meta events and system-exclusive events will not be paid attention to in the project, hence there will be no more detail in relation to these.

2.1.3 Why MIDI?

In terms of audio use and manipulation, MIDI has a number of advantages:

Firstly, it is much easier to estimate information such as beat locations, chord labels, or predominant melody from these representations than from an audio signal. This is because all the notes and timing are given with exact timing without the need of any time to frequency conversions, avoiding windowing and other kinds of distortion of the signal. In conclusion, MIDI files allow starting from a 'cleaner' material.

Secondly, the amount of available midi files is huge. Also due to the little storage space midi files take (only megabytes), more and more data is being stored and distributed in this format, implying the amount of data available will probably increase notoriously in the future.

2.1.4 Limitations of MIDI

From an artistic point of view, MIDI has some limitations such as the fact that it does not have clear explanations on how the communication should be between devices (leading to strong dependence on the knowledge of the user); and the price of devices that make use of MIDI is quite high.

From a technical point of view there are three main problems:

Firstly, MIDI clogs. MIDI is a system based on messages, and they are not as fast as they could be desired. This may lead to a big number of messages coming simultaneously, overwhelming MIDI's ability to handle these messages and therefore collapsing the system.

Secondly, MIDI lag. There could be the case that the time between the sending of a message and the execution of it results in a delay that is audible for human ears. The result can turn out to be more distracting than expected, especially if it is repeated during the musical piece.

Thirdly, a limited number of channels. Even though is it not a limitation for this project, in the case of future improvements, if an orchestra were to be recorded, this is a problem to deal with.

Despite these problems in MIDI, they will not affect notoriously this work since MIDI is not used in real time for this project, and therefore they do not have much weight in comparison to the advantages it offers.

2.1.5 Quality of MIDI

As will be mentioned further in the text, when using a chord recogniser and measuring its efficiency there is one very important variable that could influence the results significantly: the quality of the MIDI file.

What is meant by MIDI file quality?

As explained before, Midi stores the information of note values, tempos and dynamics, and to extract these from an audio file low level analysis must be done. The best methods for this kind of extractions are compiled by MIREX, and in the MIR section of this text a chart that expresses the performance of recognition systems shows that efficiency is around 61%, hence results are not extremely accurate. (Table 4 : MIR exficiency parameters extracted from MIREX).

It is concluded that there is still much work to be done and recognisers are far from accurate. Hence, when using a MIDI file in an efficiency test there is a question to be asked: where does the MIDI info come from?

The first option is having an audio recording, e.g. an MP3, and we used a MIDI converter to obtain the file. Even though there is a wide range in MIDI converter qualities where much money could be paid to obtain high quality, which ever version used, MIDI converters are based on MIR methods. By the conclusions extracted from MIREX data, not a very accurate result should be expected. This error is not insignificant and therefore should be taken into account before blaming it on the next program.

The second option is obtaining a MIDI file directly recorded from a MIDI instrument. In this case, no low-level analysis is required to extract the midi info and the data is perfectly accurate, as long as the musician has played the correct notes.

Therefore the difference between both methods is huge and should be taken into account. With the first option we are expecting a 61% of maximum accuracy while with the second option we have a 100% accuracy with the audio. This is the reason why now a days it is becoming more and more common to record music on MIDI files, it is both lighter and allows access to the exact ‘theoretical’ content of the piece. In the case of this project, it has been decided that the best option would be to record the audio while playing on a MIDI instrument. This way, both MP3 and the MIDI file of the same exact version of the piece could be obtained.

2.2 MIR Research

2.2.1 What is MIR?

As wisely stated by Casey et al. in *Content-Based Music Information Retrieval: Current Directions and Future Challenges* [11] *"Music is now a days so readily accessible in digital form that personal collections can easily exceed practical limits on the time we have to listen to them …".* With these words the concept of a technological revolution in the music world is being expressed, with the consequent need of a high level organization and analysis network to deal with this amount of available data. Music Information Retrieval is the investigation area in charge of dealing with the advances in searching, retrieving and organizing music content.

2.2.2 Approaches of MIR

In first place it is interesting to know that the intellectual background of MIR contributors is surprisingly varied. It covers from common users, to professional musicians, professional engineers, mathematicians, computer scientists or even professional psychologists. The condition of these contributors allows to divide MIR approaches into three main groups:

Table 3: High-level content descriptors [11]

High-level Description	Data Source	Task Description
Timbre	Audio	Instrument Recognition Percussive, Pitched, Ensemble Recognition
Melody / Bass	Audio / Symbolic	Melody-line extraction Bass-line extraction
Rhythm	Audio	Onset detection Meter identification Meter alignment (bars) Beat (tactus) tracking Tempo tracking Average tempo
Pitch	Audio	Single fundamental freq. Multiple fundamental freq.
Harmony	Audio / Symbolic	Chord label extraction Bass-line extraction
Key	Audio / Symbolic	Modulation tracking Pitch spelling
Structure	Audio / Symbolic	Verse / chorus extraction Repeat extraction
Lyrics	Audio	Singing detection, lyrics-identification, word recognition
Non-Western music	Audio	Micro-tonal tuning systems Non-Western canon of concepts

1) Metadata: It contains factual information such as artist, album, year of publication, track title, duration, etc. Any music consumer that wishes to share its audio file can add these metadata. On the one hand this implies an advantage since adding metadata would take much time and this is done voluntarily by common users of music. On the other hand, since no rules have been established, many incongruences and subjective opinions appear.

2) High-level Music Content Description: They could be described as musical features that professional musicians are trained to recognise: Timbre, Melody/Bass, Rhythm, Pitch, Harmony, Key, Structure, Lyrics, Non-Western Music. Below is shown a table that presents most high-level descriptions with a brief explanation of the task carried out and the type of source used for each case.

3) Low-level Audio Features: They are obtained by information contained in the digital audio and they are extracted from a digital music signal using a windowed fast Fourier transformation (FFT) as the spectral extraction step. The extracted features include Short-time Magnitude Spectrum, constant-Q/ Mel Spectrum, Pitch Class profile, onset detection, Mel/Log-Frequency Cepstral Coefficients, Spectral Flux, Decibel Scale and Tempo/Beat/meter tracking. Below is a figure that illustrates the usual procedure used when extracting low-level audio features. They all involve windowing the signal, followed by an FFT. After these two main steps, the following ones depend on the best method for each particular feature and its application.

Out of the three main groups metadata is quite limited and therefore is not of interest for investigation. However, both high-level music content descriptors and low-level audio

features are being worked on nowadays. The usual procedure is to obtain the first group based on the parameters extracted from the second.

2.2.3 Audio Analysis

A distinguishing characteristic of music in comparison to speech recognition or text IR systems is that it has several streams of related information occurring in parallel. They are organized both vertically (in frequency) and horizontally (in time). Furthermore, music information is built with hierarchical schemas. In order to analyse and search music effectively, systems must seek to represent many of these views simultaneously.

As explained before, a relevant goal in MIR is extraction of high-level music content descriptions from low-level audio processes. In the following section research into extracting high-level descriptions will be explained, oriented to transforming musical audio content into representations that are intuitive for humans to search and manipulate. The discussion starts with the first problem encountered in high-level music description: beat tracking, tempo estimation and meter tracking.

1. Beat tracking: As mentioned before, music is organized both horizontally (in time) and vertically (in frequency). In order to interpret music, beat tracking is essential. However, it involves a complex procedure and it is expressed as *"the process of organizing musical audio signals into a hierarchical beat structure"* [11]. By this, it is meant that the beat structure is initially inspected by a quarter note level (represents almost regularly spaces beat times), followed by the measure level (represents bars).

The extraction of quarter-not level is based on two parameters: period and phase. The period is inversely related to the difference in time between two beats while the phase corresponds to actual beat positions that equal to zero at beat times. To estimate these parameters onset times are used as cues. By onset it is meant *"the beginning of a musical note or other sound, in which the amplitude rises from zero to an initial peak"* [14].A very common technique is to analyse how frequent inter-onset intervals (IOI) values occur. This is based on the idea that onset times tend to coincide with beat times and therefore the temporal difference between two onset times is likely to be the beat period.

To estimate the beat period, a simple technique is to calculate the histogram of IOIs between two adjacent onset times and pick out the maximum peak. However, this is not a very accurate technique. It will probably depend on the genre of the music. Another more sophisticated method is based on a windowed autocorrelation function of an onset-time sequence, power envelope, or spectral flux of the input signal where consistently repeated onset periods will appear as peaks in the output signal. In any case, it has been suggested that for efficient audio-based beat tracking, the signal must be split into frequency bands to apply one of the previous methods to each band and then combine the results to obtain a final result far more accurate.

Despite the existence of quite accurate techniques, there is still much ambiguity to be dealt with in this area. In many situations there are various hypothesis that seem plausible. When this occurs, probabilistic models may be used or multiple hypothesis can be maintained. The estimation of measure level depends on the previous estimation and musical knowledge is essential to determine it. Moreover, music knowledge is also useful in determining ambiguous situations in the detection of beat period and investigators such as

Goto [11] are doing much research on combining both musical and engineer knowledge to progress in this direction. The beat detection issue is a relevant problem since many practical applications depend on the structuration constructed by it. This explains why the beat tracking topic still attracts many researchers and new approaches are proposed every year.

2. Melody and Bass Estimation: They are both important because melody forms the core of Western music and is a strong indicator for the identity of a musical piece while the bass is strongly related to the harmony. Despite musical analysis on stereo recordings (with several channels) facilitates the task, monaural signals are commonly found and cannot be converted back into stereo, and therefore methods on this topic are designed assuming monaural signals. In any case, separating different lines of sound in a monaural signal is still an unresolved case and results are not satisfying when analysing polyphonic recordings. This is why melody and bass estimations are only applied on music that has distinct melody and bass lines, such as popular songs. The most popular methods for melody and bass estimation are the following:

a. By finding the predominant F0 (fundamental frequency) trajectory. The first idea proposed was dividing the range of frequencies into high and middle frequencies assuming they correspond to the melody, and low frequencies assigned to the bass. Then, the trajectory of the F0 was analysed separately in both frequency ranges. This method was not accurate since the decision for the position of the frontier between both frequency ranges was not based on solid grounds. To improve this method, Goto proposed the PreFEst (Predominant-F0 Estimation method) that suggested dividing the spectrum into more than two frequency bands without assigning them to melody or bass and apply mathematical algorithms to select the best F0 trajectory. Furthermore, other researchers proposed adding another dimension: discriminating voice or non-voice. This is useful assuming the voice tends to contain the melody of the music. Results proved this actually improved the accuracy of the estimation.

b. Knowledge Based Estimation of Melody lines: Method proposed for detecting melody lines using various knowledge sources to choose the most likely succession of F=s as the melody line. It was designed especially for classical sonata or concerto, where an instrumental solo melody can go up and down a wide range of frequencies and using frequency-range limitation is not possible. This model is based on previous knowledge and uses a support vector machine (SVM) and Hidden Markov Models (HMM).

In conclusion, the state of the art of melody and bass estimation has significant difficulties in dealing with complex polyphonic sound mixtures containing sounds of various instruments and singing voices. However, despite the difficulties, technologies proposed have proven being useful in many applications and many researchers have been attracted to this topic and are therefore working on it.

3. Chord and key recognition: They are an important part of the Western music and it is useful in understanding the structure of music. In the present, the best performing chord and key recognition systems are based on the unification of recognition and smoothing into a single probabilistic framework by means of HMMs. The HMM starts from the idea that a musical performance will travel though a sequence of states. Therefore, the HMM consists of a transition matrix that represents the probability of one state moving to another, and an output model.

The transition matrix is obtained by training the chord recognition system with already analysed data. Hence, the more data for the training the more robust the system will be. Below is an example of a transition probability matrix for Beatles music. In this case, the trained data is very specific and therefore the results of the system will probably be very accurate. Hence, this procedure will be more efficient when classifying data before training it and using a transition probability matrix that fits the genre or class of the musical piece.

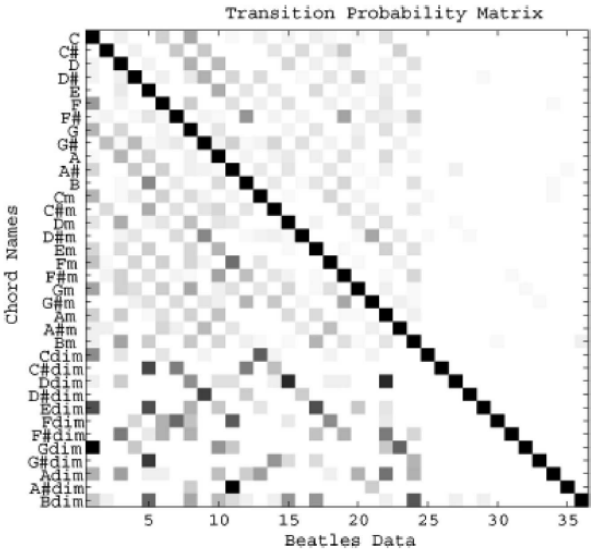

Figure 6: A 36 x 36 transition probability matrices obtained from 158 pieces of Beatles' music. [11]

4. Music Structure: This analysis consists in dividing the musical piece into regions with some internal similarity or consistency, i.e. segmenting. In this text three tasks in segmentation will be mentioned: structural analysis, smoothing, and application to Content-Based Retrieval.

Structural analysis is the task that extracts the high-level structure of a track from just the audio. In this case, knowledge on the kind of music being analysed and the kind of segment being retrieved is helpful information. Second, smoothing is a technique used to improve the previous task. It consists in smoothing over small variations to avoid them being confused with small segments in the musical piece. It is especially useful in complex music such as classical pieces. Finally, the information obtained in segmentation can be used to control the characteristics of content-based retrieval applications such as indexing homogeneous regions, allowing to save space, or in matching pieces and classifying music.

Efficiency of MIR parameters

Table 4 : MIR exficiency parameters extracted from MIREX

MIR task	**Score/ Accuracy**
Audio Tempo Estimation	0.8997
Mood Classification	0.6617
Structure Segmentation	0.6063
Audio Melody Extraction	0.6062
Audio Beat Tracking	0.5685
Audio Chord Estimation (root)	0.7903
Audio Key Detection	0.8683
Onset Detection	0.8589

The table above shows the efficiency of the best methods for some MIR tasks evaluated by MIREX in 2015. The rest of MIR tasks are attached in the annex. As seen, few tasks show very good results such as query by humming or onset tracking. The rest still have much to

improve and therefore much research is expected in these areas. For future sections of this work, it is useful to note that Midi software converters depend on these tasks. Hence, reliability on MIDI files depends enormously on whether they were directly recorded in midi format or they were converted from an audio file.

2.2.4 MIR in the future

Addressing the Semantic Gap: results and researchers opinions agree in suggesting there is a glass ceiling in performance of around 65% accuracy [11], no matter what type of system is employed. None of the models proposed appear to provide a solution for closing the gap between the acoustical and the semantic level. The conclusion is being reached that the connection between the objective measurement of audio features and the subjective description of musical emotional and perception experiences are turning out to be a hard problem. Hence, new approaches are arising in order to narrow the gap.

a. Perceptive/cognitive approach: this approach is oriented to the combination of human perception with the high semantic level of music analysis. Perception and cognition of music has a long tradition of study and has been observing musical parameters by experimentation to then model the human understanding of such musical parameters. Since human perception is subjective, this research is useful in adapting objective machine analysis to what humans actually hear or produce.

b. Emotional/Affective Approach: research on this topic suggests there is a strong correlation between emotional/affective descriptions and various parameters. More specifically, recent research has studied the relationship between low-level perceptual/acoustic features and semantic emotional/ affective concepts like mood, which seems to be a promising study line.

These final approaches represent the necessity of engineering, music and cognitive knowledge being combined and simultaneously understood in order to progress in the MIR research. This project attempts to move in that direction by combining the knowledge of the components working on it.

2.3 Connection between MIDI and MIR

2.3.1 MIR most relevant problems

As previously stated, in music features extraction, the usual procedure is to obtain high-level music content descriptors based on low-level audio features. Therefore, if the low-level audio features are not accurately extracted, the high-level descriptors will not be accurate either, since they directly depend on the previous. Time-to-frequency conversion acts as a bottleneck in this process and as long as it remains this way, improvements in the steps that follow will have little effect on the final result.

Hence, how can the time-to-frequency issue be solved? MIDI offers a number of possibilities that could imply avoiding this problem.

2.3.2 How can MIDI improve MIR efficiency?

As mentioned above, MIDI can allow avoiding the time-to-frequency matter. Nowadays, the amount of equipment and software that employ MIDI are rapidly growing and, as a result, the number of people and professionals producing music with this format is also growing. If the musical piece to be analysed and whose musical features are desired to be extracted is created in MIDI format, all the information on how the piece was played is stored: the tempo, the dynamics, and most importantly, the notes played in each instance in time. Hence, when possessing a musical piece generated in MIDI, the time-to-frequency is resolved since this step is no longer required, the frequency values in each instance in time are already known.

With this key difference, the extraction of musical features increases and problems depend on future steps, such as incongruences or multiple possible answers for a same chord.

2.4 Summary

In this chapter, State of the Art, the theory studied prior to the design and development of the idea behind this project has been explained. It is organised into two main blocks, MIDI and MIR research, and a third block in which the connection between them is shown.

In the first block, the most important information is the understanding of the format of MIDI, this will be a key point when developing the first block of design and development. Also a formal definition of MIDI is given, as well as the reasoning for choosing MIDI, its limitations and what its quality depends on. These notions are included to help comprehend what this new concept is and why it will be useful in this work.

In the second block, the approaches of Music Information Retrieval are shown, including a more detailed explanation of the approaches that are more relevant for this project. Also, possible alternatives for the future proposed by professional are briefly described. The study of MIR will be necessary for the design and development of the second part of the first block, as will be seen in the following section.

Finally, in the third block the connection between MIDI and MIR is presented, explaining how MIDI can imply solving many problems that MIR deals with. Once the knowledge required has been understood, and its use and importance for this project has been put into place, the design followed by the development of this work can begin.

3 Design and Development

In this section, the design and the development of this project will be presented and explained. The above sections dealt with introducing the main idea behind this work and showing the research the project is based on. This section is where the contributions of this work, either planned or achieved, are shown.

As presented in the Structure section, the Design and Development section is divided into three blocks: the first, development of MIDI render, aims at extracting musical features from a musical piece. It begins with a MIDI file and the desired outcome is a list of chords per half bar that represent the harmony of the musical piece contained in the MIDI file. The second block proposes a mapping of music tension-release patterns, extracted from the previous list of chords, to colour patterns. The third block presents a synchronization of coloured light, based on the colour patterns extracted in the previous block, with the music.

Below is shown a diagram explaining how the three blocks of this chapter fit together and what are the main steps in each block.

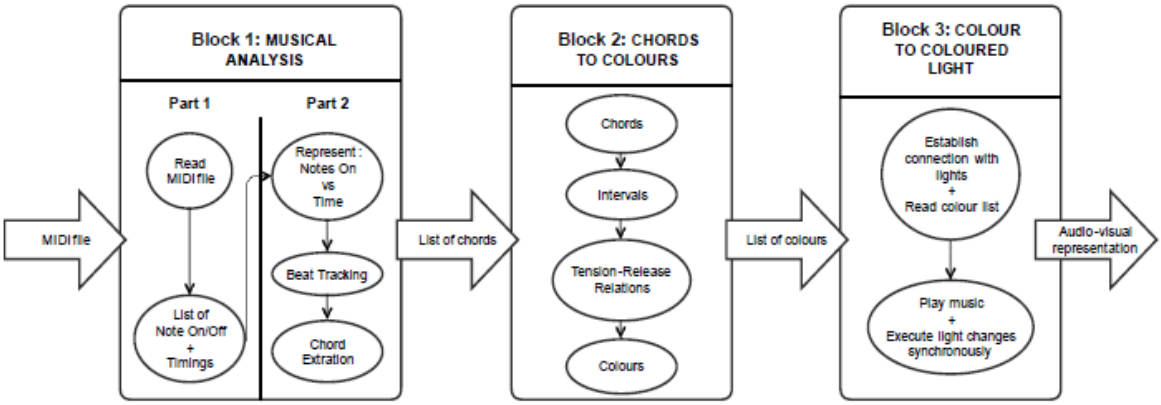

Figure 7: Overview of the three blocks contained in the design and development section and the connection between them

3.1 Musical Analysis

In this section the first block of development and design will be explained. The starting point of this block is the MIDI file of a musical piece and the objective of this block is to obtain the chords of the later musical piece. To break this problem into smaller steps it will be decomposed from back to front: starting with the aim (chords) and going backwards to the starting point (midi file).

1. Why is a chord a chord? How is a chord defined?

'A chord is any harmonic set of three or more notes that is heard as if sounding simultaneously' [14]. Chords are extracted by analysing the harmony of the piece in observation. Harmony is defined as the use of simultaneous notes or pitches and it is referred to as the 'vertical' aspect of music. The study of harmony involves chord

construction, chord progressions and the principles of connection that govern chords. Only the chord construction is relevant for this section, but the others will be useful to interpret relations between chords in following sections [14]. In conclusion, to define a chord in a given interval of time, harmony knowledge and the simultaneous notes in that time are required. However, there is another matter to be taken into account:

2. How is the time interval defined?

Which notes are taken into account to determine a chord and when do they start being part of the next chord? This is naturally defined by rhythm. Music is rhythmically divided into beats and beats are grouped into bars. In order to understand the harmony of a piece it is crucial to detect its rhythm. This parameter is usually contained in the midi file expressed in the form of beats: either PPQ or STMP, explained in the state of the art section. Once the duration of a beat is known, the number of beats observed in order to detect a chord depends on the application: usually the music genre gives a hint on the velocity with which harmony varies. In the case of this project, to simplify the problem and concentrate on a specific case only classical music has been analysed and the duration of a chord has been established to one or two chords per bar. However, this parameter could be easily modified if required.

3. Conclusion

To sum up, to reach the objective of this section, the proposed steps to achieve it are the following:

1. Extract the notes or pitch sounding at every moment in time

2. Divide the list of notes into notes per beat and note per bar

3. Group notes into chords following harmony rules.

Before executing any of the previous steps, a relevant decision had to be taken: which audio format should be used to analyse a music piece? Daily used formats such as wav or mp3 were considered, but MIDI files were chosen as the best option. The reasons for this decision are explained in the state of the art section.

3.1.1 Part 1: Reading of MIDI file in Processing

The first part of the first block consists on reading the MIDI file and extracting two values of the notes contained in it: their pitch and the moment in time in which they sound.

After thorough search online and consulting staff members at Tue, the conclusion was reached that there are no Matlab programs or toolboxes capable of reading and interpreting MIDI files. Hence, it was decided that Processing, a programming language based on java language, would be employed to generate a program that reads and interprets MIDI files: Midirender. This choice was mainly motivated by the existing libraries contained in javascript for midi that facilitate the extraction of data in comparison to C++. The use of javascript classes and the MidiBus help avoiding simple 1 bit mistakes that lead to drastic errors in the rest of analysis as could happen in C++ programming.

In order to read the binary code of a MIDI file, a very meticulous comprehension of its format had to be made: a thorough understanding of the structure and organization of header and trunk chunks. A better look at the format of MIDI files is explained on the 'State of the Art' section. The information shown in the tables Table 1 : MIDI header chunk format and Table 2: MIDI track chunk format are used to carry out this section.

Also, a basic knowledge on Java is necessary to develop this program. Here is a schematic explanation of the classes and functions employed in the project:

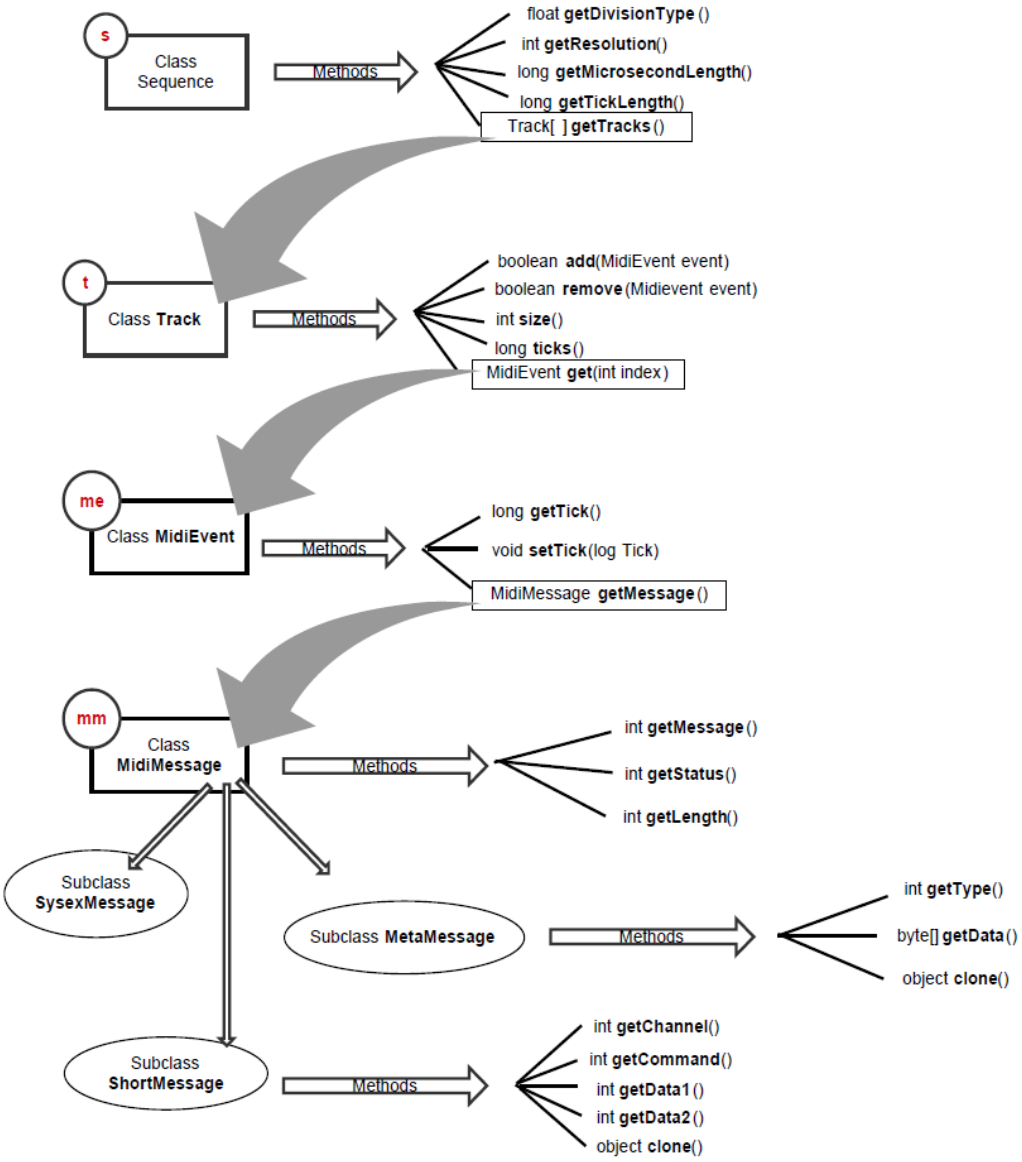

Figure 8 : Schematic representations of clases employed to generate midi render

Firstly, the sequence is extracted and the whole code is dumped into the variable *s*. Secondly, the information from the header and the tracks are extracted from the sequence. The information extracted from the header is the division type (PPQ or SMPT), the duration of the sequence in microseconds and in ticks, and the resolution (ticks per beat in

case of PPQ and ticks per frame in case of SMPT). This data is written onto the output file *data.csv*.

Once the general information for the whole file is extracted, the tracks are interpreted. For every track, *t*, all the events extracted onto the variable *me* and its message is extracted onto the variable *mm* by means of the function getMessage. Then, the command is extracted from the message: if it is NOTE ON, a 1 is printed on the output file, if it is NOTE OFF, a 0 is printed on the output file. The pitch and tick are also extracted from the message of every event and also printed on the output file. When the program has been run, a csv file is generated with all the relevant information of the file in a format that Matlab can interpret.

In summary, the outcome of the processing program is a list of instructions for notes to turn on and off. Each line has the instruction turn on/ turn off, a frequency value given in a range from 0 to 127 (where 60 is the middle C), and a timestamp that allows to place each note in a time scale. The timestamp is expressed as a function of a delta timing, which represents a fraction of a beat. The delta timing is a number stored in the header of the midi file that defines how many ticks there are in a quarter beat (semiquaver), where the tick is the unit in midi files. Once this information is extracted and represented in a manageable format for Matlab, the csv is opened by a Matlab program, where better processing and representation of the information can be made.

3.1.2 Part 2: Interpretation of MIDI notes in Matlab

In this section of the MIDI render the information a MIDI file holds is already comprehensible for Matlab. The next step is to structure and organize the long list of notes going on and off so they can be interpreted into chords.

In first place, some general information of the musical piece is required. Some is extracted from the csv file: format, number of chunks and delta timing; and other is expected from the user. Visualization parameters and information of the bar type can be very helpful for the program and require very little effort from the user, especially if we are talking about a person with musical knowledge. The definitions and main concepts concerned with parameters mentioned above are previously explained in the state of the art section.

To represent the data in a visual way, a blank matrix was defined and each frequency value was filled with black in the moments in time when they were expressed as NOTE ON on the list. The outcome of this procedure is shown below. The figure shows five bars of the piece of Claire de Lune. The black areas represent the pressed notes in terms of midi beats and their corresponding frequency value (expressed from 0 to 127) is represented on the y axis.

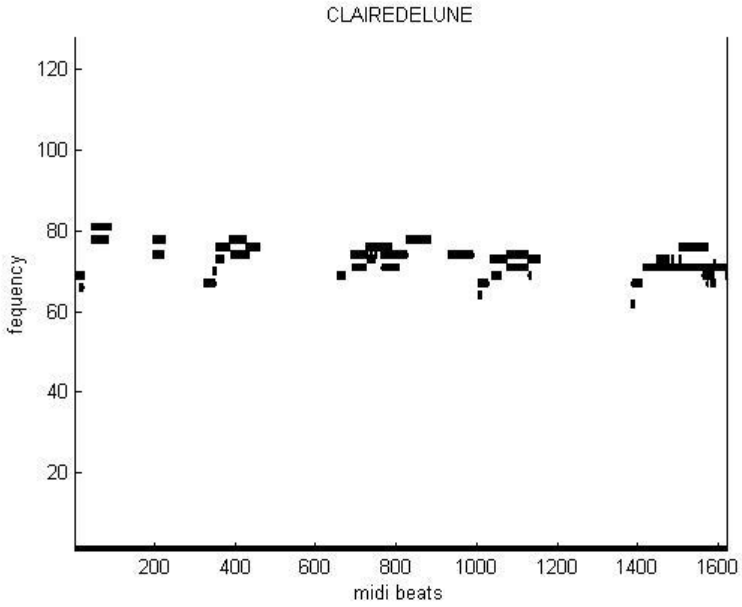

Figure 9: Visual representation of frequency per MIDI beats of Claire de Lune

Once the midi file could be visualised and could be contrasted with a music sheet as proof that the notes made sense, a new component was added: bars.

Bars and timing

Once the concept of delta timing is understood most of the work has already been done. If the delta timing is the number of ticks per quarter note (being tick the unit in MIDI files) and the time signature is directly related to the number notes there are in a bar, the number of ticks per bar is delta timing x time signature. (Using a relation chart compass type-number of notes per bar).

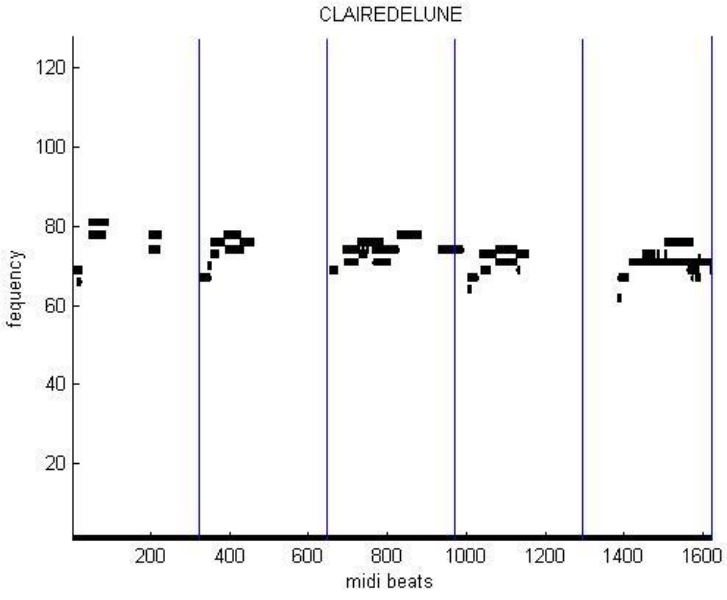

Figure 10 : Visual representation of frequency per MIDI beats of Claire de Lune with bars

Above is shown the result of representing the extracted bars together with the previous representation. Even though it might not seem very inaccurate, the truth is the bars don't fall exactly where they should according to the music sheet. Despite music sheets indicating exact timings, music is played by humans who are no metronome. Hence, behind matching bars into a music piece there are many studies involving probability and statistics taking into account which characteristics have in common, the beginning or the end of a bar, 'strong' note, etc. More about research related to this topic is explained on the state of the art section.

Since extracting the information for chord detection directly didn't seem possible, a substitute was proposed: a MIDI chord detector.

3.1.3 Part 3: Find Midi Chords

To overcome the bars and timing issues from the previous part, FindMidiChords was used to substitute the initial procedure. As expressed by the author, it is "a program to analyse and display the chords within any MIDI file" [12]. The outcome of this software was then used to continue with the rest of the project. Hence, with this software the objective of obtaining the chords of the musical piece in analysis is achieved.

Explanation of functioning

After reading articles explaining several chord detection or tonality detection procedures mostly based on MIR, it would have been logical to expect a functioning of the program based on Hidden Marcov Models or other similar probabilistic models. However, to our surprise this program does not use probabilistic methods or training. Unlike other methods, it is designed to *"smooth out"* the chords across the defined interval. One may believe by smooth out it is referred to filtering notes depending on their intensity to take into account the attack-sustain-decay-release curve. However, it has nothing to do with this either. By smoothing out it is meant to directly eliminate notes of short duration, which are probably passing notes and not part of any chord.

After the smoothing out procedure and having the information prepared, the chord detection starts. First of all, notes are represented as numbers from 1 to 11. Below is an example where A is the main tonality.

Table 5 : Correspondance of american nomenglature music notes to representative numbers

1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**
A	Bb	B	C	Db	D	Eb	E	F	G	Ab

The main tonality can be extracted from the MIDI file, in many cases there is a Key Specification containing this data. Otherwise, statistics on how many times each note is played are used and matched against a note row for each key.

Thus if notes C, D E, F, G, A and B are the most used, this would indicate the key of C, or possibly the related minor (Am). It could also indicate a modal scale which uses the notes of C (E.g. Dorian mode on D, Myxolydian mode on G), but those are not searched for, as most users would probably not understand.

Once notes are referred to as numbers relative to the main tonality, the software establishes which notes are being played simultaneously (depending on a parameter defined by the user that decides how many chords per bar). These are matched against the possible chords by means of internal tables devised by the author of the software. It is concluded that the author has made use of musical knowledge, either of his own or consulting a professional musician.

In the tests and results section, chapter 4, the reliability of this software is evaluated. To do so, a method based on speaker diarization is used, where the outcome of this chord-extraction software and compared with the musical analysis of a professional pianist.

Block 2

3.2 Chords to musical tension to colour

Once the midi file has been interpreted into a list of chords in the previous block, the next step is to translate musical tensions into colour tensions. That is the aim of this block.

In order to achieve this, the first step is to express the chords as relative distances between them, called intervals. The next step is to find changes in tonality or in modulation during the piece. Finally, there is a relationship established between the 'jumps' in harmony and the emotion they evoke with the emotion associated to a colour.

3.2.1 Chords to intervals

The program chords to colour mapping is in charge of achieving the objective mentioned above: establishing a relationship between music tension and colour tension. To start, the chords file extracted from FindMidiChords is initially moulded into a more appropriate shape, turning a file full of useless characters and extra information into a clean array of chords. The layout has been established that a line represents a bar, to resemble the organization of a music sheet. By default, FindMidiChords defines four beats per bar and it has been decided to show one chord per beat. A sample of the result is shown below, where there are four chords per line that stand for a bar and for four beats.

```
                MIDI Chords 2.0.1
                from TRANTOR Ltd
              Phone UK (+44) 0 1639-633072

      Chords in MIDI file C:\mchord\CLAIREDELUNE.mid

Main notes in tune  C  Db Eb F  Gb Ab Bb
Probable Key is D flat Major or related minor
Initial Time signature is 4/4
Using chords: Major&7th, minor&7th, Maj7
        dim, aug (+5), sus4, 9, 11, 13
        and 'slash' chords, eg Cm/D
Chords in bar:  4
 Including  Channel  1  (Acoustic Grand Piano)

Bar
  1 | (F)     Fm      Db      =       |
  2 | .       .       .       .       |
  3 | Bbm     Eb7     Bbm     (F)     |
  4 | .       (Db)    =       .       |
  5 | Ab      Bbm     .       .       |
  6 | .       .       .       .       |
  7 | (Bb)    (C)     (Bb)    Eb      |
  8 | Ebsus4  (Bb)    Ebsus4  Ebm     |
  9 | Ab      (Gb)    .       .       |
 10 | .       .       (Ab)    (F)     |
 11 | (Gb)    (Bb)    Bb      (Eb)    |
 12 | .       (Eb)    (Db)    (Eb)    |
 13 | .       .       .       .       |
 14 | .       .       .       .       |
 15 | (Db)    Dbmaj7sus4 Fm   =       |
 16 | (Ab)    .       Db      .       |
 17 | .       (Gb)    .       Gbm6    |
 18 | (Eb)    .       .       .       |
 19 | .       .       (C)     Db      |
 20 | (F)     Db      =       .       |
 21 | Db      Fm      .       (Gb)    |
 22 | .       Gbm6    (F)     .       |
 23 | .       .       (Db)    .       |
 24 | .       .       Ab      Db      |
 25 | (Eb)    Bbm     =       (Ab)    |
 26 | .       .       .       .       |
 27 | Bb      Eb      (F)     (Eb)    |
 28 | .       (Db)    =       .       |
 29 | .       (Bb)    .       .       |
 30 | .       .       .       Bb      |
 31 | .       .       (Eb)    =       |
 32 | Eb      Bbm     (Db)    (Bb)    |
 33 | (C)     (Db)    (Bb)    .       |
 34 | .       Bb      .       =       |
```

Figure 11 : Exaple of how FindMidiChords softsare outputs the extracted chords of the mucial piece

Table 6 : Example results of notes extracted from FindMidiChords document

	Beat 1	**Beat 2**	**Beat 3**	**Beat 4**
Bar 1	'F'	'Fm'	'Db'	'Db'
Bar 2	'Db'	'Db'	'Db'	'Db'
Bar 3	'Bbm'	'Eb7'	'Bbm'	'F'
Bar 4	'F'	'Db'	'Db'	'Db'
Bar 5	'Ab'	'Bbm'	'Bbm'	'Bbm'

Once the data is adapted to a manageable format, it can be examined. Even though the layout on the table above can be interpreted by any human and understood by any musician, letters are still an uncomfortable format. This is the reason for remoulding the file one more time. A brief justification for the way in which it will be remoulded is done below.

It is important to understand that music works with distances between chords, or the so called tensions. As long as the tension or distance between chords is maintained, the whole musical piece can be shifted upwards or downwards in pitch and it will still sound well (the same melody will be interpreted). Hence, it is concluded that the name or the pitch of a chord is not as important as the way it relates to the chords amongst it.

It is also important to understand the difference meant by the concepts distance, tension and interval in this text. On the one hand, when talking about tension in music it is referred to the emotion evoked in the listener, to how strong is the urge felt for a chord to fall into a more stable one. Hence, it could be expressed as the perceptual component of the distance between chords. On the other hand, the interval is the difference between two pitches expressed in half tones. It could be expressed as the mathematical component of the distance between chords, a number.

Despite musical tension being a matter of interest for further comparisons with colour, this section is focused on the rational side and distances will be referred to in terms of intervals. As explained before, all the relevant information from the list of chords is contained in the distance and therefore it can be represented as a number. This translation of chord names to distance numbers can be done in various ways: distance from previous chord, distance from first chord, distance from defined chord, etc., and therefore, various possibilities have been designed. Below are explained the different methods used, in form of functions, and the resulting pattern for each case.

1.1 Succession A

This function translates the list of chords into a list of intervals in reference to the chord A. The mapping pattern for the intervals would be the following, where the first row shows the interval number assigned to each chord in the second row:

Table 7 : Format of chords after going throught function 'Succession A', when reference chord is A

1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**
A	Bb	B	C	Db	D	Eb	E	F	G	Ab

Even though the tonic of the chord (the name we give the chord) is the most important, there are other parameters or information contained in a chord that are relevant mostly for the decision of change in tonality. These are stored in a matrix together with the interval number. The information kept is whether the chord is major or minor, if the chord is maj7 and if it is diminished. Other information is not relevant for this project and will therefore be omitted. Below are shown the possible options for each column and a sample of the first chords. Below these options, a table with an example for this case is also shown.

% first column: jump in half tones from A to the chord

% second column: 1 == major chord // 0 == minor chord

% third column: 0 == maj7 chord// 1 == not maj7 chord

% fourth column: 0 == dim chord// 1 == not dim chord

Table 8 : Example of chords and corresponding value for each situation

Chord name	**Jump**	**Major/Minor**	**Maj7**	**dim**
'F'	9	1	1	1
'Fm'	9	0	1	1
'Db'	5	1	1	1
'Db'	5	1	1	1
'Db'	5	1	1	1
'Db'	5	1	1	1
'Db'	5	1	1	1
'Db'	5	1	1	1
'Bbm'	2	0	1	1
'Eb7'	7	1	1	1
'Bbm'	2	0	1	1
'F'	9	1	1	1
'F'	9	1	1	1
'Db'	5	1	1	1
…				
'Dbmaj7sus4’	5	1	0	1
…				
'Gbdim'	10	0	1	0

1.2 Succession B

This function uses the returned list of the previous function SuccessionA to translate the chords into intervals in reference to the first chord of the list instead of the chord A, which was chosen randomly.

In comparison to the outcome of successionA, only the first column varies. Below is shown the beginning of the matrix created.

Table 9: Format of chords after going throught function 'Succession B'

Chord name	Jump	Major/Minor	Maj7	dim
'F'	0	1	1	1
'Fm'	0	0	1	1
'Db'	8	1	1	1
'Db'	8	1	1	1
'Db'	8	1	1	1
'Db'	8	1	1	1
'Db'	8	1	1	1
'Db'	8	1	1	1
'Bbm'	5	0	1	1
'Eb7'	10	1	1	1
'Bbm'	8	0	1	1
'F'	0	1	1	1
'F'	0	1	1	1
'Db'	8	1	1	1
…				
'Dbmaj7sus4’	8	1	0	1
…				
'Gbdim'	1	0	1	0

1.3 Succession C

This function also uses the returned list of the function SuccessionA, this time to translate the chords into intervals in reference to the previous chord of the list.

The outcome only contains the jumps. Since a difference from one chord to the next is expressed, it no longer makes sense to include additional information. What additional information should be included? The previous or the next?

Table 10: Format of chords after going throught function 'Succession C'

0	-4	0	0	0	0	0	-3	5	-5	7	0	-4	0	0	-5	2

This list is one unit shorter than the previous ones since the first chord is not included because it cannot represent a jump without a previous chord. Another notable difference with the previous successions is that negative numbers appear. Before, numbers were expressed in a circular scale from 1 to 12 where 14 would be 3 again. In this case, chords can move in both directions. All of these possibilities are simply different ways of representing the same. For further work, Succession B will be used because it was thought to be better adapted for programming the next phases. However, the others could be used if preferred.

1.4 Succession D

This function uses the returned list of the previous function SuccsessionC to translate the numerical intervals into musical intervals, i.e. the nomenclature musicians use to express those intervals. An array with the names used in music and a pattern guide were declared to match both components.

This was generated for better communication with colleague Katarina and other musicians. It is not indispensable for the project, however, it allows checking work with professionals in middle stages, which is useful.

Table 11: Format of chords after going throught function 'Succession D'

First chord: F	**Jump (SuccC)**	**Musical Interval**	**Ascendant/ Descendent**
'Fm'	0	unison	
'Db'	-4	mayorThird	descendent
'Db'	0	unison	
'Db'	0	unison	
'Db'	0	unison	
'Db'	0	unison	
'Db'	0	unison	
'Bbm'	-3	minorThird	descendent
'Eb7'	5	perfectFourth	ascendant
'Bbm'	-5	perfectFourth	descendent
'F'	7	perfectFifth	ascendant

Below is presented the correspondence between the way musicians refer to intervals and how they are referred to in the program to develop the results of SuccessionD.

Table 12: Correspondence of half-step intervals to musical nomenclature

Half-Steps	**Scale Degree**	**Degree Name**	**Interval Name**
0	1	Root or Tonic	Unison
1	b2	Flat second	Minor second
2	2	Second	Major second
3	b3	Flat third	Minor third
4	3	Third	Major third
5	4	Fourth	Perfect fourth
6	b5	Flat fifth	Diminished fifth
7	5	Fifth	Perfect fifth
8	#5	Sharp fifth	Augmented fifth
9	6	Sixth	Major sixth
10	B7	Flat seventh	Minor seventh
11	7	Seventh	Major seventh
12	8	Octave	Perfect octave

3.2.2 Chord or tonality?

In this last part two options are possible: either changing the colours with every chord, or changing the colours with every change in tonality. Whether to trust one option or the other is up to the user. Knowing which modality would be more convenient requires some musical understanding and generally slow pieces will be better suited by chord changes while more dynamic pieces will appreciate tonality changes since chord changes would result as a saturating effect and aim of the lights would be damaged. However, this matter is out of the subject of study of this project and therefore, for the time being, the user will be trusted for this aspect. In the future, this could be an issue to be studied.

If the option 'change with every chord' is selected, the list of chords does not need any more modifications. If the other option is chosen, the problem is how to decide whether a change in tonality has taken place or not. Even though a basic knowledge in music is helpful in understanding this process, it is not the main subject of this project and therefore the procedure for taking this decision has been reduced into simple steps provided by a professional pianist.

If the three conditions expressed below occur, it is considered as a change in tonality.

1. Interval = + 4 (4th ascendant in musical terms)

2. First chord is major or diminished

3. Second chord is major or minor but not maj7

3.2.3 Musical tension to colour tension

The last part of this section is the mapping of musical intervals or tonality changes onto a number that represents a chord. This outcome is printed onto a document to be opened with processing and interpreted into colours of light.

The choice of colour for each musical tension depends on a study carried out by Katarina Biljman [7]. This study goes through the following steps, as illustrated in the figure bellow:

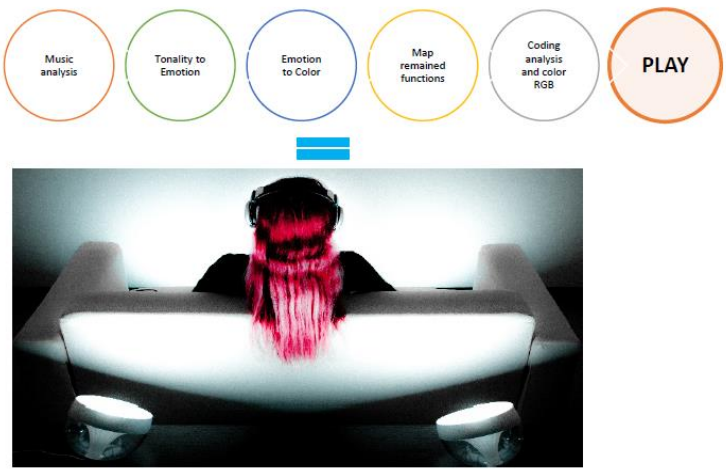

Figure 12: The music-to-colour translation process [7]

After the musical analysis, tonality is translated to emotion. Research [1] has proved the emotions awakened by music are related to harmonic tensions, hence Katarina has based on this to match these [7].

In a parallel way, the same as musical tensions are associated to emotions, numerous artists have investigated to world of colour and emotion. Despite non-professionals being able to match red to passion, black to sad or yellow to happy, studies have established more strict associations based on proof and experiments [2].

Once this information is gathered, an association between colour and music tension can be made, leaving the emotions for the audience to experience. Below, is the result of this matching my Katarina:

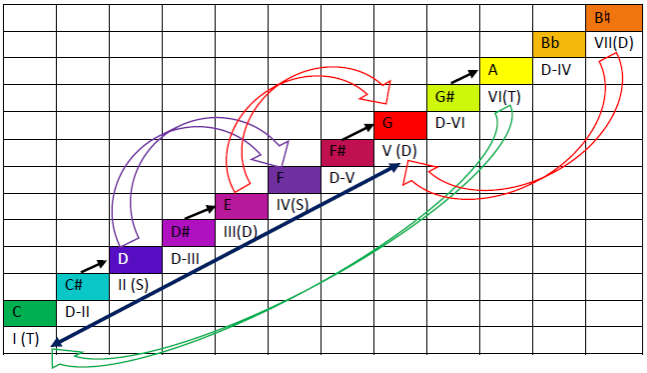

Figure 13: A musical scale coloured according to the mapping principle [7]

Below, an example where the tonic is assigned to red and dominant (maximum tension) is assigned to the complementary colour of red: green.

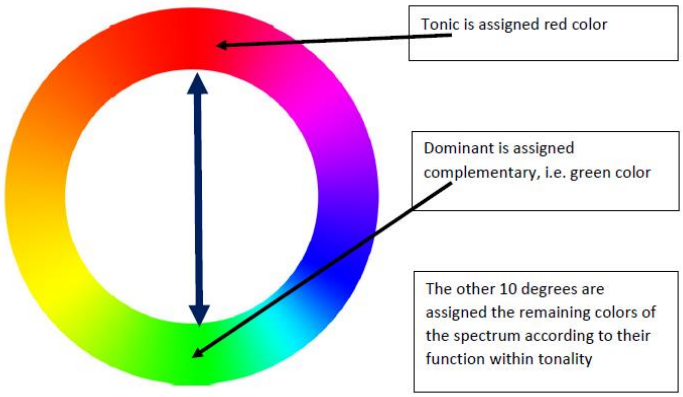

Figure 14: An example of the principle of the used mapping scheme, shown on a colour spectrum [7]

Block 3

3.3 Colour to coloured light

3.3.1 Design

The objective of this section is to execute the commands contained in a list of colours as changes in the colour of lamps. To achieve this, in first place, Phillips HUE lamps were selected. They have an extended range of varied colours and they are controlled via Wi-Fi, which allow controlling them without touching, and therefore avoid spoiling their effect. In second place, the environment was chosen: Processing, where javascript is used. This decision was taken because there was access to a selection of functions developed with this program which allow establishing connection and controlling the HUE lamps [13]. This assured the connection with Processing was possible and the connection establishment would not start from cero, hence this choice was the best option.

Once those choices were made, the router and bridge to the lights would have to be installed and configured, and private access to internet be requested.

Afterwards, the list of colours will be read, the music will be played, and the lights will be commanded to change to the correct colour in the correct time. To execute the command in the correct moment, a wait function will be employed between change and change.

3.3.2 Development

This section is responsible for sending the change in colour signal to the lamps. Hue lamps from Phillips Company were chosen due to their broad range in colours and their connection via Wi-Fi. This type of communication allows more mobility and flexibility of its use. Moreover, other members of the investigation group in TUe that had already been using these lights expressed their happiness with these lights [13]. This research group also provided the project with a model script in order to communicate lights, router and computer or mobile application. The functioning of this script will be explained later in this section.

As mentioned previously in the text, this part of the program is written in processing using java script language. Some manipulation of the script developed by the investigation group were made in order to adapt it to the list of colour changes obtained in the previous section.

The design steps were followed as expressed in the design section. Java functions and functions obtained from an investigation group [13] were used to establish connection with the lamps, to load and play the music, to read line by line the colour list file, and to execute the changes in colours. The colours were defined in HSV and could be modified as desired by the user. Nevertheless, there was a complication: colours were expressed in bars and the music file is expressed in time. Even though using the delta time contained in the MIDI file and the time signature, the time a bar lasts could be calculated, music does not strictly maintain the theoretic duration of the bars and therefore synchronization of colours and music was not possible. For synchronization to occur, beat tracking should be used, however, this will be included in the future work section and a temporary solution is proposed in the next section.

3.3.3 Hue Program Trial Lighting

Since the previous program had synchronization unresolved problems due to rhythm inconsistency in human-played music, another parallel program was developed as an alternative. After all, the aim of this project was not just investigating but also handing in a useful and 'correctly running' program that can improve the efficiency of demonstrations such as concerts or recordings. Hence, this could be expressed as a second version of the previous program. The modifications were the following:

In relation to synchronization, fixing this issue requires way more time than the available, as well as knowledge on probabilistic data and Marcov models, which are needed to resolve this issue as accurately as possible. This was also previously mentioned in the first block when matching bars to the harmony in the correct places was not an easy job. Therefore, what was the solution? Since the changes in colours were already listed and the question was finding the exact moment in time to change them, this had to be done manually. The program gave an estimation of these times, and a professional pianist had to modify them by ear. A list of timings was written, which corresponded to the changes in colour, and the program read both lists.

After the manual synchronization step, the rest was executed the same way as the previous script. The result of the use of this script is reflected in the recorded demo video of the project, explained in the next chapter, Tests and Results. Also, it is being used for further investigation and recordings of results.

3.4 Summary

In the design and development section, the plan in order to achieve the objectives proposed in the introduction section have been presented and how they were carried out was explained. This section was divided into three blocks, where the justification behind the choice of this division is found in the programming environment employed in each block.

Firstly, the Development of MIDI render block starts with a MIDI file, which is read and interpreted into notes in time by a program developed in Programing, in javascript language. Afterwards, a musical analysis is proposed with the aim of representing the notes into chords per half bar. The later part involved too many timing problems and therefore, an external software was proposed to achieve the objectives of the first block and enable moving on to the next block.

Secondly, a mapping from musical tension-release patterns to complementary-colour patterns is proposed. Initially, relations between chords are expressed as relative tensions, referred to as intervals. Then, the musical intervals are mapped onto colour intervals following research findings [7], producing as a final result a list of colours.

Finally, colours are visually expressed in the form of lights which ideally change synchronously with the music. This type of synchronization is highly sophisticated and requires complex beat tracking algorithms which exceed work load of this work. Therefore, an alternative program was designed and developed which allowed introducing synchronization timings manually, in order to achieve the goal of this block. As a result of this alternative program, the outcome was managed to be recorded and is included as an annex to this work. To evaluate and testify the efficiency of the work proposed and carried out in this section, a series of tests and results were made and are shown in the following section Integration, Tests and Results.

4 Tests and Results

4.1 FindMidiChords reliability

When depending on an external software, it is wise to test the accuracy of its results. In order to do this, the results extracted by FindMidiChords software were compared to a 'correct' analysis, i.e. the musical analysis of the pieces carried out by a professional pianist. In this section, the method employed to carry out this evaluation will be explained and then, the results will be presented and commented.

4.1.1 Chord Diarization

To evaluate the software, the speaker diarization method is adapted to this particular situation, and this new variant will be referred to as chord diarization. First, the differences and similarities between both types of diarization will be expressed, to demonstrate the coherence of adapting speaker diarization to chord diarization.

1. Definition

Gerald Friedland states in the book *Speech and Audio Processing: Perception of Speech and Music "the goal of speaker diarization is to segment a single or multi-channel audio recording into speaker-homogeneous regions with the goal of answering the question who spoke when?"[15]* The goal of chord diarization is to segment a single channel music audio into half-bar sections with the goal of answering the question what chord was being played when?

To sum up, Friedland proposes two questions to resolve the diarization problem: *"What are the speech regions? Which speech regions belong to the same speaker?"* In the table below, the correspondence to those questions for chord diarization is proposed. The answers to the chord diarization questions are also answered.

Table 13: Chord diarization questions and answers compared to speaker diarization questions

Speaker diarization questions	Chord diarization questions	Chord diarization answers
What are the speech regions?	What are the chord regions?	Every region is half a bar
Which speech regions belong to the same speaker?	Which chord regions (half-bars) belong to which chord?	Every software will offer its best answer to this question.

2. Steps

Friedland explains speaker diarization systems follow three tasks: *"First, discriminate between speech and non-speech regions, second, detect speaker changes to segment the audio data, third, group the segmented regions together into speaker-homogeneous clusters." [15]*

In chord diarization, it is assumed there are no silence regions, hence the first step is not necessary. The second step that involves detecting changes, in chord diarization is defined as half a bar by default. Finally, the third step in chord diarization carried out by repeating a chord when it is maintained during various half-bars and therefore there homogenization is not done either.

3. The output

"The output of a speaker diarization system consists of labels describing speech segments in terms of start time, end time, and speaker cluster time." [15]. The output of a chord diarization system consists of the same, where start time and end time are pre-defined and are measured in half bars instead of seconds, and instead of assigning a speaker cluster, a chord cluster is assigned. Despite these conceptual modifications, the system works the same way in both cases.

How is the output computed?

A standard metric that measures the accuracy of speaker diarization has been defined by the US National Institute of Standards and Technology (NIST). The systems that follow this standard evaluate the output against manually-annotated ground-truth segments, usually with timings refined through ASR forced-alignment. The two segmentations are compared by using a dynamic programming procedure to find the optimal one-to-one mapping between the hypothesis and the ground truth segments so that the total overlap between reference speaker and the corresponding mapped hypothesized speaker cluster is maximised. Below there is figure showing how it works:

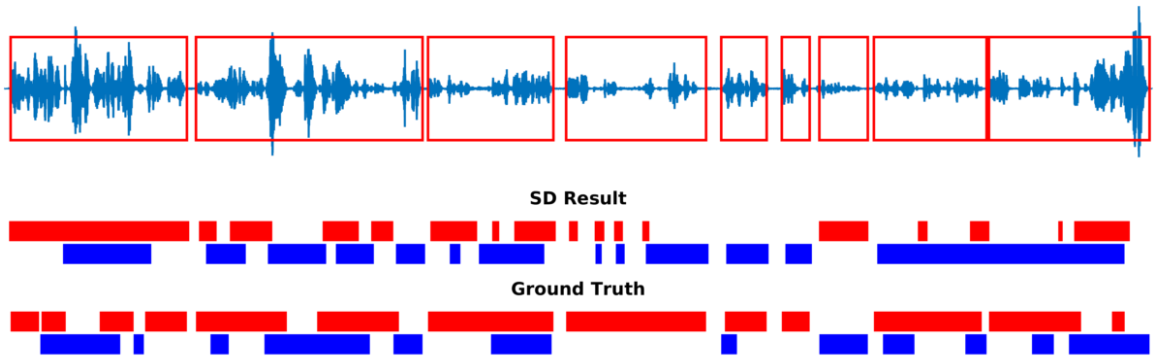

Figure 15: visual representation of how speech diarization works. From an audio file, results are extracted, then they are aligned and compared to the ground truth. [16]

The difference between the speaker diarization result and the ground truth is expressed as Diarization Error Rate and is defined in the following way:

Equation 1: Diarization Error Rate definition

$$DER = \frac{\sum_{s=1}^{S} \text{dur}(s) \cdot (\max(N_{ref}(s), N_{hyp}(s)) - N_{correct}(s))}{\sum_{s=1}^{S} \text{dur}(s) \cdot N_{ref}}$$

Where S is the total number of time segments, Nref and Nsys are the number of speakers speaking in segment s, and Ncorrects indicates the number of speakers that speak in

segment s that have been correctly matched. DER is usually expressed in percent, hence, when all speakers are matched correctly, the error rate is 0%, and therefore, the task of speaker diarization is minimizing the equation above.

In the case of chord diarization the speech diarization is represeted as the chords extracted by the software being evaluated, and the timings are expressed in terms of half bars. The ground truth is given by a professional pianist who has manually extracted the chords from the music sheet corresponding to the audio. The pianist was told to extract chords per half bar and this way the timings will always match. As well as in speaker diarization, chord diarization aims at obtaining a DET percentage as low as possible.

4.1.2 Results

To obtain data that shows the efficiency of FindMidiChords software, both results extracted from the software and the analysis given by the pianist are manipulated so they are aligned. To do this, time signature is taken into account to include the same number of beats per bar, and number of chords per bar are also checked so they match. Moreover, in music, various names for the same type of chord are accepted, as well as a note can be expressed in to ways, e.g. F# sharp is the same note as Gb. These different nomenclatures are unified to a common one to avoid the software diagnosing wrong chords when they are simply named in a different way.

Once the ground truth and the chords extracted by the software are expressed in the same way, they are analysed using chord diarization, i.e. speaker diarization adapted to chords. The method used is explained in the previous section. Below are shown the results of some classical pieces that have been compared, as a percentage of errors.

Table 14: overall speaker diarization error for FindMidiChords results compared to Ground Truth

Musical piece	Overall speaker diarization error
Claire de Lune, Debussy	26.21%
Op.70 nº1, Chopin	13.37%
Op 69 nº2, Chopin	53.65%
Op 69 nº2 trio Chopin	23.21%
Mean	29.11%
Giuliette as a little girl, Prokafiev	51.30%

As explained in the previous section, the lower the DET (Diarization Error Rate), the better the system is. In the table of results it is seen that the mean DET of the results for the first four musical pieces analysed is 29.11 %. The amount of data is not enough to prove any hypothesis, however, it can give an idea of the accuracy of the results extracted by the software used.

In the *Table 4 : MIR exficiency parameters extracted from MIREX* shown in the State of the Art section, it is stated that the best audio beat tracking is 56.85% and best audio chord estimation (root) is 79.03%. In comparison to these values, 29.11% DET, which implies a

mean 71.89% correct answers, is a decent efficiency value, considering the previously commented results evaluated by MIREX. It must be taken into account that these results were obtained by MIDI files directly recorded for MIDI instruments. However, the last musical piece, Romeo and Giuliette, was analysed by using a MIDI file converted from an mp3 file. As expected, the results for this piece are not as accurate as the mean of the rest, being this an example of the previously stated theory that the quality of a MIDI depends on whether it was directly recorded, or converted from an audio file. However, the efficiency of this for this file was not dramatically worse than the others, and therefore more files should be analysed to conclude there is actually a big difference.

Other software, *MIRtoolbox* [17] and *KeyDetection* [18] were examined to test whether FindMidiChords was a good choice. However, they did not use beat tracking and would not divide into bars or beats and therefore the alignment with the Ground Truth were not possible. It was concluded that if the beat tracking was not possible, the results could not be better than FindMidiChords.

It was observed that the software may have problems in the extraction if the timing signature is extracted wrongly. In the case of Claire de Lune, the MIDI file was initially analysed as 4/4 signature and results were very poor. However, another midi file was used, in which the time signature was correctly used, 9/8. It is concluded that when analysing musical pieces with more complex time signatures, as in the case of Claire de Lune, results can be extracted poorly if the time signature is not exacted correctly.

4.2 Demo video

To visually represent the results obtained from this project, two demo videos were recorded. In the videos, a classical piano piece is played and the lights change following the changes in harmony of the later musical piece. Two different musical pieces were chosen: Claire de Lune composed by Debussy, in which the changes are slow; and a sonatina composed by Mozart, in which the changes are quite agile. Recording both videos was useful to compare the number of chords per bar each piece requires depending on its tempo and how accurate synchronization was in each case. The conclusion taken from the previous comparison was that the Claire de Lune video seemed to contain more accurate changes. The reasons behind this are deduced to be due to its slower tempo and therefore slower harmony changes.

In the demo videos, it can be appreciated how vivid, tense moments in music lead to red, bright lights while calm moments involve lights changing to cold colours such as blue or purple. Although this distribution of colours was agreed to be used as explained for the demo videos, other colours can be assigned to tense and release musical moments, the only requirement imposed is that the tense and release colours are complementary. Moreover, when choosing the colours, a compromise between obtaining visible changes and producing an agreeable experience had to be taken into account.

To send the change of colour signals to the lights, the Hue Program Trial Lighting was employed. Hence, the timings were manually given the pianist. This implied numerous modifications of seconds to the left or to the right, yet synchronization was not as accurate as desired in certain moments. Nevertheless, the result is representative of the ideal final result, and accomplishes the functions and expectations of this section.

4.3 Summary

In this section, an evaluation of the design and development outcomes has been made.

Initially, an efficiency test of the external software FindMidiChords employed in the first block was made. In order to evaluate how well the chords were extracted, they were compared to other software results based on audio files instead of MIDI files and they were all contrasted with the analysis a professional pianist provided. The technique used to compare these results was based on diarization, an evaluation method employed in many other signal processing areas. Results are shown in a table and commented.

Secondly, a demo video is included, which visually explains the results achieved. In this video, lights change with the music played and it can be appreciated how the vividness or tension in the audio can be seen in the form of more 'tense' colours such as red or pink.

The results on both sections prove there is still much to improve in this work. Nevertheless, they also prove having achieved the objectives initially proposed even if the procedure could have been executed in a more elegant way on some occasions. These conclusions are more broadly dealt with in the following section Conclusions and Future work.

5 Conclusions and Future Work

5.1 Conclusions

To express the conclusions, the objectives are refreshed below, to comment whether they have been achieved correctly or not. Objectives:

- Automatic extraction of harmony information (chords and change in tonality) from a classical musical piece contained in a midi file.
- Automatic mapping of music tension-release patterns to colour
- Automatic synchronization of coloured light changes with music harmony changes
- Evaluation of software employed performance

These objectives were aimed to fulfil both the end of bachelor project expectations and cover the requirements of Katarina Biljman's team to complete the technological side of her doctoral thesis. This might have led to proposing objectives too ambitious for the amount of work expected for this kind of project and therefore the tasks were not executed as flawlessly as desired. Nevertheless, they were all completed and the objectives were achieved.

The first objective, automatic extraction of harmony information, was designed to be achieved by programming it, but the program developed did not produce results as accurately as required and hence an external software was used to cover the leaks of this section. The conclusion extracted from this first objective is that even though the final result was not obtained in the initially proposed way, an alternative was found and the objective was achieved.

The second objective, automatic mapping of music tension-release patterns to colour, was successfully achieved. The program was developed following research findings, and they were approved of by Katarina.

The third objective, automatic synchronization of coloured light changes with music harmony changes, did not result as it was initially designed either. To do so, it involved beat tracking which was too complex. Nevertheless, an alternative program was developed to adapt to what the pianist could easily include manually and the problem was solved. Hence, once more, even though the final result was not obtained as initially proposed, an alternative solution was proposed and the objective was achieved.

Finally, the fourth objective, evaluation of software employed performance, was also achieved. The results are shown in the fourth chapter and they prove the software employed gives good enough results. Even though not much data was employed, the mean values extracted is a DET of 29.11% which is considered as a good enough number, considering other efficiency results gathered by MIREX (Table 4 : MIR exficiency parameters extracted from MIREX).

As a whole, the expectations of the project were fulfilled. Technological knowledge acquired during the career and by doing research was employed to make possible a musical project. It is believed that the two disciplines were combined in a proper way succeeding to create something new and innovative. Generating the ideal product that could be actually used by a professional music therapist goes beyond what can be achieved with the work of a final bachelor project. However, with the work presented in this text, a solid base has been constructed, hence completing the first step towards achieving a broader goal. This implies many improvements should be made on the previous work in order to move another step closer towards the product desired. These improvements and ideas intended to be carried out in the future are included in the following section, Future work.

5.2 Future work

As explained in the Conclusions section, this project has achieved its goals, however, there are still many things that could be modified or included to improve this project and get closer to a bigger goal: generating a product that can be used by professionals. The possible changes or improvements proposed are explained below.

5.2.1 Time and synchronization

This matter has already been mentioned along the text in various occasions. To be able to synchronise music with chords or music with colours, both must be measured in the same unit: beats. Hence, the problem lies in how to figure the length of the beat in a musical piece, and how to follow the beat along the piece, since music is never metrically exact.

Firstly, in order to carry out a musical analysis, determining the tempo and bars of the piece is crucial. Hence, to improve the first block of the Design and Development section and be able to generate a program that could estimate chords per bars and therefore stop depending on the FindMidiChords external software, dealing with beat tracking would be necessary. Secondly, the same information is required in the last block of the Design and Development section, synchronization of coloured light with music harmony. In order to change the colours on the correct timings, it is necessary to have and algorithm than is able to break a musical piece into bars. Researchers [11] are working on this matter, most of them based on hidden marcov models and probabilistic patterns. If these studies were analysed, the outcome could be used to improve these sections.

5.2.2 MIDI advances

This section is related to how the quality of MIDI varies notoriously from being converted from an audio file to being directly recorded from a midi instrument. More and more files are being recorded in midi format, increasing the amount of useful material for programs that depend on midi files, such as the one proposed in this project. Therefore this is not an improvement that depends on the author of this work, but a condition that affects this product and that is believed that will change for the better in the future.

5.2.3 Chord detection

Another improvement proposed is developing a proper chord detection program so external software is no longer necessary. In order to achieve this, a deeper research should be done about this topic. Initially, as mentioned in the first proposal of Future work, beat tracking should be done. Then, most probably, HMMs would be used since now a days it is

the most commonly used method. Moreover, applying musical theory would also be practical in achieving this improvement.

5.2.4 User interface

Taking into account that long-term objective of this project is producing a product for non-technological professionals, offering a friendly user interface will highly desirable to improve its value. Even though this proposal could considered as a detail for the last steps towards achieving the product, will be a relevant change when it is the right moment to be done.

5.2.5 Music style extension

Whether chord change, tonality change, or modulation change should be the best variable for the coloured lights to change, depends primarily on the music style of the piece in observation. This is the reason why classical music was chosen as the only musical style observed in this project. This way the type of change can always be the same. However, it would be an interesting approach to adapt this program to other musical styles. To do this, music genre detection is proposed. This type of detection is being investigated as part of Music Information Retrieval research. The results of these detectors are relatively efficient as shown on the *Table 4* : MIR exficiency parameters extracted from MIREX in which MIREX results of efficiency parameters are shown.

5.2.6 Software unification

In order to improve efficiency and generate a better organised product, it would be desirable to unify the various programs developed in this project into a single complete program that executes all parts. This program would be developed in Processing employing javascript language, since MIDI files cannot be read by Matlab and the HUE lamps can neither be connected through Matlab.

5.3 Summary

In the section of Conclusions and Future Work, an evaluation of the whole project and the fulfilment of the objectives has been made, followed by a number of proposals for improvement in the future.

The general conclusion reached is that all objectives have been achieved, however, in the attempt of proposing a project compatible with university requirements and demands of the Sound Design team in the Technical University of Eindhoven, objectives proposed might have resulted excessive and some adaptations to the original designs had to be done in order to achieve them. Nevertheless, as expressed above, despite not obtaining a flawless product, the results desired were obtained and the objectives desired were fulfilled. In the Future work section, improvements were proposed to generate a more stable and consistent project in the future, with the objective of producing a product that can be used by a professional music therapist.

References

[1] Wigram, T., Pedersen, I. N., & Bonde, L. O. (2002). *A comprehensive guide to music therapy.* Jessica Kingsley Publishers.

[2] Valdez, P., & Mehrabian, A. (1994). Effects of Color on Emotions. *Journal of Experimental Psychology*.

[3] K., B., J.H., E., M., D., & M, F. (2014). Music Visualisation with Colored Ambiance Lights. *Experiencing light 2014 International Conference on the Effects of Light on Wellbeing*.

[4] K., B., A., C., M., D., J.H., E., & M, F. (2015). Sensing the Music – Designing multisensory therapeutic environments with music and light. *Eindhoven: ILIAD15 Proceedings public release*.

[5] Betancourt, M. (2015). Making Music With Color: How color organs connect light and sound. *The Atlantic*.

[6] Collopy, F. (2000). Color, Form, and Motion: Dimensions of a Musical Art of Light. *Leonardo*.

[7] Biljman, K. (2015). *Mapping representation.*

[8] Rothstein, J. (1995). *MIDI a comprehensive introduction.* A-R Editions.

[9] Selfridge-Field, E. (1997). *Beyond MIDI.* The MIT Press.

[10] John, C. (5 de August de 2004). *Steganography V - Hiding Messages in MIDI Songs.* Obtenido de http://www.codeproject.com/Articles/5390/Steganography-V-Hiding-Messages-in-MIDI-Songs

[11] Casey, M. A., Veltkamp, R., Leman, M., Goto, M., Rhodes, C., & Slaney, M. (2008). Content-Based Music Information Retrieval: Current Directions and Future Challenges. *Proceeding of the IEEE*.

[12] Jennings, N. (s.f.). *Find Midi Chords Software description*

[13] Offermans, S. (2015)

[14] *https://en.wikipedia.org*. (s.f.)

[15] Friedland, G. (2011). Speaker Diarization. En B. Gold, N. Morgan, & D. Ellis, *Speech and Audio Signal Processing: Processing and Perception of Speech and Music* (págs. 644-653). John Wiley & Sons, Inc

[16] Gebru, I. D., Ba, S., Li, X., & Horaud, R. P. (s.f.). *Audio-Visual Speaker Diarization Based on Spatiotemporal Bayesian Fusion*. Obtenido de https://team.inria.fr/perception/research/avdiarization/

[17] Lartillot, O. (7 de December de 2014). MIRtoolbox. Aalborg University, Denmark: http://www.jyu.fi/hum/laitokset/musiikki/en/research/coe/materials/mirtoolbox.

[18] Lerch, A. (s.f.). KeyDetection. *Matlab sources accompanying the book 'An Introduction to Audio Content Analysis - Applications in Signal Processing and Music Informatics'*. http://www.audiocontentanalysis.org/code/.

Glossary

ASR	Automatic Speech Recognition
DER	Diarization Error Rate
MIR	Music Information Retrieval
MIDI	Musical Instrument Digital Interface
MThd	MIDI header chunk
MTrk	MIDI track chunk
NIST	National Institute of Standards and Technology
PPQ	Pulses Per Quarter Note
SMPTE	Society of Motion Picture and Television timecode