Comenius University in Bratislava
Faculty of Mathematics, Physics and Informatics

Dynamic visualization of classical music using analytical metadata

Bachelor Thesis

2024
Maximilián Martin Krošlák

Comenius University in Bratislava
Faculty of Mathematics, Physics and Informatics

Dynamic visualization of classical music using analytical metadata

Bachelor Thesis

Study Programme:	Computer Science
Field of Study:	Computer Science
Department:	Department of Computer Science
Supervisor:	doc. RNDr. Andrej Ferko, PhD.

Bratislava, 2024
Maximilián Martin Krošlák

75941259

Univerzita Komenského v Bratislave
Fakulta matematiky, fyziky a informatiky

ZADANIE ZÁVEREČNEJ PRÁCE

Meno a priezvisko študenta:	Maximilián Martin Krošlák
Študijný program:	informatika (Jednoodborové štúdium, bakalársky I. st., denná forma)
Študijný odbor:	informatika
Typ záverečnej práce:	bakalárska
Jazyk záverečnej práce:	anglický
Sekundárny jazyk:	slovenský

Názov: Dynamic multi-view visualization of classical music using analytic metadata
Dynamická viacpohľadová vizualizácia klasickej hudby s využitím analytických metadát

Anotácia: Predstavujeme vybrané riešenia pre vizualizáciu klasickej hudby pomocou priestorových 3D modelov a navrhujeme ich integráciu do viacpohľadovej vizualizácie podľa Browna.

Cieľ: 1. Prehľad problematiky v dostupnej literatúre.
2. Špecifikácia projektu.
3. Implementácia v prostredí Blender a experimenty.

Literatúra: PURWINS, H. et al. Toroidal Models in Tonal Theory and Pitch Class Analysis. In Tonal Theory for the Digital Age (Computing in Musicology 15, 2007), 73–98. [online] https://www.ccarh.org/publications/books/cm/vol/15/cm15-05-purwins.pdf
FERKOVÁ, E. – ŠUKOLA, M. A Software for Analysis of Harmony as a Computer Tool for Search of Tonal Cadencies in Various Midi Files. DMRN + 14: Digital Music Research Network One-day Workshop 2019.
TYMOCZKO, D. Why topology? Journal of Mathematics and Music, 2020. [online] https://doi.org/10.1080/17459737.2020.1799563
BROWN, M. H. 1988. Algorithm Animation. MIT Press. ISBN: 9780262524117.

Kľúčové slová: Midi, vizualizácia, Blender, hudba, viacpohľadovosť

Vedúci:	doc. RNDr. Andrej Ferko, PhD.
Katedra:	FMFI.KAG - Katedra algebry a geometrie
Vedúci katedry:	doc. RNDr. Pavel Chalmoviansky, PhD.
Dátum zadania:	13.10.2023

Dátum schválenia: 16.10.2023 doc. RNDr. Dana Pardubská, CSc.
garant študijného programu

....

študent vedúci práce

75941259

Comenius University Bratislava
Faculty of Mathematics, Physics and Informatics

THESIS ASSIGNMENT

Name and Surname:	Maximilián Martin Krošlák
Study programme:	Computer Science (Single degree study, bachelor I. deg., full time form)
Field of Study:	Computer Science
Type of Thesis:	Bachelor´s thesis
Language of Thesis:	English
Secondary language:	Slovak

Title: Dynamic multi-view visualization of classical music using analytic metadata

Annotation: We present selected solutions for visualizing classical music using spatial 3D models and propose their integration into multi-view visualization according to Brown.

Aim:
1. Research findings in relevant literature.
2. Project specification.
3. Blender implementation and experiments.

Literature: PURWINS, H. et al. Toroidal Models in Tonal Theory and Pitch Class Analysis. In Tonal Theory for the Digital Age (Computing in Musicology 15, 2007), 73–98. [online]
https://www.ccarh.org/publications/books/cm/vol/15/cm15-05-purwins.pdf
FERKOVÁ, E. – ŠUKOLA, M. A Software for Analysis of Harmony as a Computer Tool for Search of Tonal Cadencies in Various Midi Files. DMRN+ 14: Digital Music Research Network One-day Workshop 2019.
TYMOCZKO, D. Why topology? Journal of Mathematics and Music, 2020. [online] https://doi.org/10.1080/17459737.2020.1799563
BROWN, M. H. 1988. Algorithm Animation. MIT Press. ISBN: 9780262524117.

Keywords: Midi, visualization, Blender, music, multi-view

Supervisor:	doc. RNDr. Andrej Ferko, PhD.
Department:	FMFI.KAG - Department of Algebra and Geometry
Head of department:	doc. RNDr. Pavel Chalmovianský, PhD.
Assigned:	13.10.2023

Approved: 16.10.2023 doc. RNDr. Dana Pardubská, CSc.
Guarantor of Study Programme

....

Student Supervisor

Acknowledgments: I would like to express my deepest gratitude to my supervisor, doc. RNDr. Andrej Ferko, PhD., for his invaluable guidance, support, and encouragement throughout this project. his expertise and insightful feedback have been instrumental in shaping this thesis. I am also thankful to my family for their support and understanding. Their encouragement and belief in me have been a constant source of motivation.

Abstrakt

Táto práca prezentuje vybrané riešenia vizualizácie klasickej hudby pomocou priestorových 2D a 3D modelov a navrhuje ich integráciu do viacpohľadovej vizualizácie podľa Browna. Vybrané hudobné dáta z Midi súboru vizualizujeme rôznymi technikami. Dynamická vizualizácia sa vykonáva v softvéri Blender, hlavne pomocou doplnku Animation nodes.

Kľúčové slová: Midi, vizualizácia, Blender, hudba, viacpohľadovosť

Abstract

This thesis presents selected solutions for visualizing classical music using spatial 2D and 3D models and propose their integration into multi-view visualization according to Brown. We are visualizing selected musical data from the MIDI file using various techniques. The dynamic visualization is done in the software Blender, mainly using the Animation nodes add-on.

Keywords: Midi, visualization, Blender, music, multi-view

Contents

List of Figures

Introduction

The goal of this thesis is to create a multi-view visualization of classical music using spatial 2D and 3D models. This thesis is not only for musicians interested in music visualization, but is intended to be read by anybody who is even a tiny bit interested in how can music look like if we do not hear it, but see it. The music is meant to be enjoyed by everybody, be it professional musicians or even a little child with no musical knowledge.

This thesis is divided into five chapters. In the first chapter, we will in short introduce the history of music, from the initial primitive musical instrument, to the first attempts of writing down music and all the way to the present, where music is a part of our everyday life. At the end of this chapter, we will mention MIDI, which revolutionized the way music is created and distributed.

In the second chapter, we will focus on visualization. No, not visualization as a technique or practice of imagining what you want to achieve in the future, which directs your subconscious to be aware of the end goal you have in mind. We will introduce scientific visualization as a technique of creating images, diagrams or animations to communicate a message. Visualization today has ever-expanding applications in science, education, engineering, and medicine but mostly in computer graphics. If you did not meet with visualisations of information in the appearance of graphs, then you definitely must have seen a map of a city or a world map. That is a form of visualization, too. But this thesis will mainly focus on something more 'abstract'. And that is a visualization of music.

In the third chapter, we will mention selected approaches for visualizations of music. If you are a musician or have an interest in music, you probably have heard about Tonnetz or Circle of Fifths and if you have not, you definitely must have seen a musical notation. That too is a form of visualization of music. But in this chapter, we will focus on some more complex ways to visualize music, which is not only limited to two-dimensional space, but some are even in three dimensions. We will only introduce them and offer some pictures, but if you want to know more I recommend reading the referenced books.

In the fourth chapter, we will describe the entire process of implementation of our work. This thesis is heavily inspired by Brown's BALSA software [8], which is software that is used to study and visualize algorithms. But what makes it special is its multi-view visualization (more about BALSA is in chapter 3). This thesis is about multi-view visualization of music. We take some of the visualizations mentioned in Chapter 3 and some of our own original visualizations and put them together to create a multi-view visualization of different views and perspective on music. This should hopefully help musicians to easier analyze music, or for curious kids to pass the time.

In the fifth chapter, we will talk about problems that we encountered during the process of creating this thesis and the results of our work whether we were successful or not and mention plans for the future for improvement of our work.

Chapter 1

History of music

1.1 Music in the past

Music has been a part of human civilisation since the distant past. It first arose in the Paleolithic period, also known as the Old Stone Age, by evidence of bone flute [19]. Early humans communicated through vocalization, percussive sounds or chants. These primal forms of music were used in rituals and communal bonding, to create a collective sense of belonging within ancient societies.

In antiquity, at the advent of writing, music flourished mainly in literate countries like China, Greece, Rome, Egypt and Mesopotamia. Each country developed its unique musical traditions and instruments, reflecting its social structure and beliefs. Egyptians composed hymns and chants dedicated to their deities, Mesopotamians crafted lyres and harps, while Greek philosophers (like Pythagoras) focused more on mathematical principles [17] in musical harmony, which later influenced Western music theory. Even though the music of ancient societies was diverse, in some aspects, they were similar. For example the use of mono-phony, improvisation and dominance of text in musical settings.

The medieval ages witnessed the rise of sacred music, mainly played in churches during sermons, liturgical rituals, ceremonies and worship of God. Gregorian chants [20], distinctive by their monastic simplicity and solemnity, presented medieval musical aesthetics. It beautifully reflected the religious, cultural, and social dynamics of European society during this period.

The Renaissance was the golden age of choral music. Composers employed complex music harmonies and counterpoint techniques to enrich musical textures, expressive melodies, and lush harmonies. The invention of printing had an immense influence on the dissemination of musical styles and contributed to the establishment of the first international style in European music. Most commonly found in households were lutes and keyboard instruments [4].

1.2 Present

In the 19th century, music sheets restricted access to music only to middle to high-class people who were literate and owned musical instruments. Later, radio broadcasting [24] and mass market availability of gramophone records became the main way to learn about new songs and musical pieces, allowing low-income people to listen to music, too.

The 20th century saw a revolution in music as it witnessed unprecedented innovations and experimentation. World War I influenced many of the arts, including music, and composers began exploring darker, harsher sounds. The advent of recording technology enabled the mass dissemination of recordings [1] and music consumption.

In the digital age, music has become easily accessible and more diverse than ever before. New media and technologies were developed to record and edit music. The globalization of music has led to cross-cultural exchanges and hybrid genres, making the lines between traditional and contemporary styles indistinct. The musical landscape continues to constantly evolve, reflecting the complexities of modern society and human experience.

1.3 MIDI

In modern music production and performance, the Musical Instrument Digital Interface, commonly known as MIDI, stands as a cornerstone technology that has revolutionized the way musicians create, record, and manipulate music. MIDI is a technical standard that describes a data communication protocol, digital interface and electrical connectors which enable electronic musical instruments, computers and other devices to exchange information and control signals among each other [22].

The significance of MIDI is that it permits a wide variety of musical equipment from different manufacturers to cooperate in a single system. Each piece of equipment adheres to the same rules of information management specified by MIDI so that they have a common system of communication. MIDI devices communicate with each other over MIDI channels. They receive data transmission in real-time, allowing for live performance or recording. Interacting with a MIDI device by tapping a key or button will be converted into a MIDI event, which represents specific instruction, such as note, pitch, volume or timing. All these events are saved into the MIDI file, which does not contain actual audio wave-forms like mp3 audio files, but instead, they contain these instructions and musical events, such as notes, instrument settings (which musical instrument is playing), tempo changes, velocity, control messages and many more [2].

Everything encoded in MIDI data format specification is called MIDI message [22]. Each message describes a particular event, such as the start of a musical note, its duration, pitch and others. MIDI symbolically describes the electronic steps required to generate the sound. We can think of it as MIDI specifying exactly which electronic circuits should turn on and how long they should remain on. Because MIDI files do not contain actual audio they are relatively small compared to mp3 or WAV audio files. They are highly editable and flexible, allowing for changing note pitches, altering music instrumental settings or adjusting timing and tempo.

The MIDI devices need to carry out two conditions to be MIDI devices. First, the hardware interface and the latter the data format. The hardware needs a MIDI port, which converts MIDI data to the electrical voltages representing the MIDI data and second, the MIDI cable for transmitting the voltage signals from one MIDI port to another, which converts the voltages back to digital (MIDI) data [22]. Every MIDI device must contain at least one microprocessor to interpret the MIDI messages and act on them. MIDI devices also have a MIDI In and MIDI Out jack, but also a third jack called MIDI Thru, which records (makes a copy) of all incoming MIDI messages through the MIDI In port. This allows for easy recording of musical pieces, which can later be modified to suit the musician's expectations. MIDI offers composers and producers superb control and flexibility over their compositions. MIDI data allows musicians to capture and edit performances with precision, layering multiple tracks to create complex arrangements. Moreover, MIDI controllers such as MIDI keyboards, drum pads and guitars provide interfaces for musicians to interact with their MIDI workstations, narrowing the gap between musical performance and digital production.

MIDI applications have revolutionized the way music is created and performed. MIDI offers a versatile toolkit for musicians and producers to express themselves through music.

Chapter 2

Visualization

2.1 What is visualization?

Visualization is a technique for graphical representation of information or data. It transforms symbolic notation into a geometrical and coloured one. We can visualize information by using simple lines or using complex geometrical models or colours. Visualization is more than just mere aesthetics. It should serve as a cognitive amplifier for the human visual system's capability to process and derive meanings from visual patterns.

The human brain is adept at processing visual information efficiently. According to [23], visualization aids human cognition not because images or animations are superior to other forms of presenting information, but because visualization helps the user by making the world outside the mind a resource for thought in a specific way. Visualization amplifies cognition by increasing the memory and processing resources available to the user, reducing the search for information and using visual representations to enhance the detection of patterns. But at the same time, in my opinion, visualization can sometimes constrain our thinking, since it influences or even pushes our thinking in a specific way. That way it is important to find the best method or technique to visualize information correctly.

2.1.1 Types of visualizations

There exist diverse techniques in information visualization, each catering to a specific data type. From static visualizations like bar charts, graphs or tables to dynamic visualizations such as temperature or wind maps. Static visualization refers to a method of displaying data focusing on a single data relationship. Contrary to this, dynamic visualization is a method of displaying data that is constantly changing in a period, like, for example, a graph of the human population.

2.2 Visualization of music

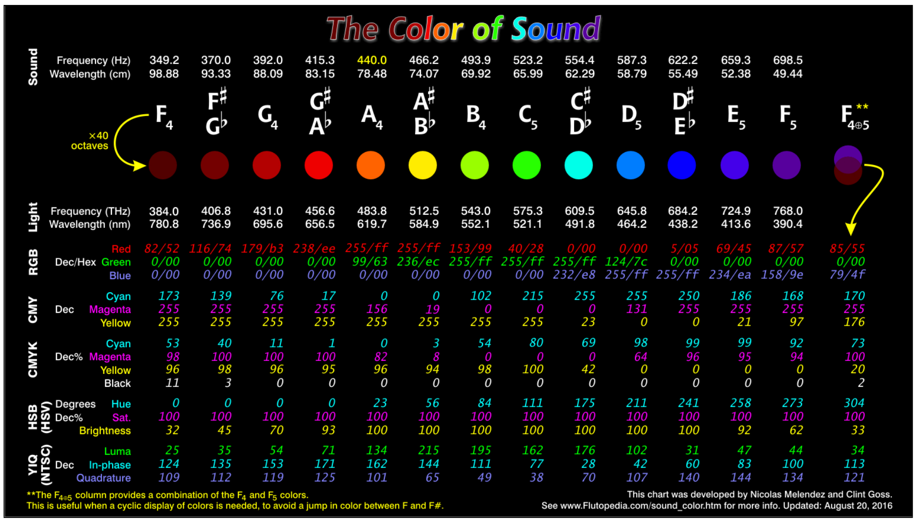
The phenomenon of synesthesia provides a foundational framework for understanding the inherent connection between auditory and visual experiences. Neuroimaging studies have revealed the intricate neural pathways that contribute to synesthetic responses, offering insights into how auditory stimuli can evoke vivid visual perceptions [11]. People with synesthesia usually perceive information by even unrelated senses. For example, they can link letters to colours or even taste words. And this is the aim of our project, based on selected ideas. To perceive music not only by our ears but also by our eyes.

Visualization of music is the transformation of sound using special methods into images or animations. It allows us not only to hear music but also to see it. Music, or in general, sound, is a vibration that propagates as an acoustic wave through a transmission medium such as gas, solid or liquid. Sound is the reception of such waves by ears and their perception by the brain. Humans are capable of recognizing sound waves that have frequencies between 20 Hz and 20 kHz.

Light too is a type of vibration, but at a much higher frequency. The visible light spectrum is vibrating on frequencies between 375 THz and 750 THz. And since sound is a vibration too, we can try and map different tones to colours. If we augment each different tone (a-b-c-d-e-f-g) by 40 octaves, we can "naturally" map tones to their respective colours [14]. For those who do not know what an octave is, it is the interval between one musical pitch and another with double or half its frequency. For example, if one note has a frequency of 440 Hz, the note one octave above is at 880 Hz, and the note one octave below is at 220 Hz. The ratio of frequencies of two notes an octave apart is, therefore, 2:1. So if we augmented the "A" note, which has a frequency of 440 Hz, by 40 octaves, we would get a frequency of 483.8 THz, which is equivalent to a frequency of orange colour in the visible light spectrum. The whole note to the colour schema is represented in Figure 2.1 [14].

Of course, visualizing music only by colours may not be enough. Music is more complex than it seems. Other important aspects are music harmony, rhythm, tempo, velocity and many others.

Šimon Prokop, in his work [26], created a 3D visualizing model of musical harmony. Each note had a corresponding box, which changed its size and spatial position depending on the note or accord played and velocity. He chose not to use colours in his visualisation, but to clearly show the change in musical key played, he proposed using rotation, which can drastically and dramatically change the appearance of the design.



The Color of Sound

		F_4	$F^\sharp$ $G^\flat$	G_4	$G^\sharp$ $A^\flat$	A_4	$A^\sharp$ $B^\flat$	B_4
Sound	Frequency (Hz)	349.2	370.0	392.0	415.3	440.0	466.2	493.9
	Wavelength (cm)	98.88	93.33	88.09	83.15	78.48	74.07	69.92
		●	●	●	●	●	●	●
Light	Frequency (THz)	384.0	406.8	431.0	456.6	483.8	512.5	543.0
	Wavelength (nm)	780.8	736.9	695.6	656.5	619.7	584.9	552.1
RGB Dec/Hex	Red	82/52	116/74	179/b3	238/ee	255/ff	255/ff	153/99
	Green	0/00	0/00	0/00	0/00	99/63	236/ec	255/ff
	Blue	0/00	0/00	0/00	0/00	0/00	0/00	0/00
CMY Dec	Cyan	173	139	76	17	0	0	102
	Magenta	255	255	255	255	156	19	0
	Yellow	255	255	255	255	255	255	255
CMYK Dec%	Cyan	53	40	11	1	0	3	54
	Magenta	98	100	100	100	82	8	0
	Yellow	96	98	96	95	96	94	98
	Black	11	3	0	0	0	0	0
HSB (HSV) Degrees	Hue	0	0	0	0	23	56	84
Dec%	Sat.	100	100	100	100	100	100	100
	Brightness	32	45	70	93	100	100	100
YIQ (NTSC) Dec	Luma	25	35	54	71	134	215	195
	In-phase	124	135	153	171	162	144	111
	Quadrature	109	112	119	125	101	65	49

		C_5	$C^\sharp$ $D^\flat$	D_5	$D^\sharp$ $E^\flat$	E_5	F_5	$F_{4\oplus5}$**
Sound	Frequency (Hz)	523.2	554.4	587.3	622.2	659.3	698.5	
	Wavelength (cm)	65.99	62.29	58.79	55.49	52.38	49.44	
		●	●	●	●	●	●	●
Light	Frequency (THz)	575.3	609.5	645.8	684.2	724.9	768.0	
	Wavelength (nm)	521.1	491.8	464.2	438.2	413.6	390.4	
RGB Dec/Hex	Red	40/28	0/00	0/00	5/05	69/45	87/57	85/55
	Green	255/ff	255/ff	124/7c	0/00	0/00	0/00	0/00
	Blue	0/00	232/e8	255/ff	255/ff	234/ea	158/9e	79/4f
CMY Dec	Cyan	215	255	255	250	186	168	170
	Magenta	0	0	131	255	255	255	255
	Yellow	255	23	0	0	21	97	176
CMYK Dec%	Cyan	80	69	98	99	99	92	73
	Magenta	0	0	64	96	95	94	100
	Yellow	100	42	0	0	0	0	20
	Black	0	0	0	0	0	0	2
HSB (HSV) Degrees	Hue	111	175	211	241	258	273	304
Dec%	Sat.	100	100	100	100	100	100	100
	Brightness	100	100	100	100	92	62	33
YIQ (NTSC) Dec	Luma	162	176	102	31	47	44	34
	In-phase	77	28	42	60	83	100	113
	Quadrature	38	70	107	140	144	134	121

**The $F_{4\oplus5}$ column provides a combination of the F_4 and F_5 colors.
This is useful when a cyclic display of colors is needed, to avoid a jump in color between F and F#.

This chart was developed by Nicolas Melendez and Clint Goss.
See www.Flutopedia.com/sound_color.htm for more info. Updated: August 20, 2016

Figure 2.1: The "color" of sound [14]

Another work was done by Dmitri Tymoczko [25], where he visualized voice leadings using topology. We will come back to this work and some others in Chapter 3.

2.3 The earliest visualizations of music

2.3.1 Notation

Currently, the most popular and practical way to visualize music is musical notation. Musical notation is any system used to visually represent music, using a complex system of symbols and marks that enables musicians to communicate and interpret musical ideas. Skilled musicians can gain an impression of how a particular piece of music sounds simply by looking at the notes on paper. This act is known as reading music.

The origins of musical notation date back to times of ancient civilizations such as Mesopotamia and ancient Greece, where rudimentary symbols were used to represent pitch and rhythm, but none of them were particularly comprehensive, which led to today's limited comprehension of their music. In the Middle Ages, the Christian Church began notating plainchant (chants used in liturgies) melodies so that the same chant could be used throughout the church. This was before the invention of five-staff notation. The notation was in the form of neumes, which are symbols above the text of the Gregorian chants to indicate melodic contours [7]. Over time, neumes under constant development evolved into the modern system of staff notation, which was created during the Renaissance period and provided musicians with a standardized method for representing pitch and duration.

Modern music notation is used all over the world by musicians of many different musical genres. Music notation consists of several essential components, each having its distinct purpose in convening musical information. The staff consist of five parallel horizontal lines, which act as a framework upon which pitches are indicated by placing oval note heads on the staff lines, between the staff lines or above and below the staff. If the note is placed too much below or above the staff, they use a ledger line to better emphasise which note it is. A ledger line is a line slightly longer than the note head and it is drawn parallel to the staff, above or below, spaced at the same distance as the lines within the staff.

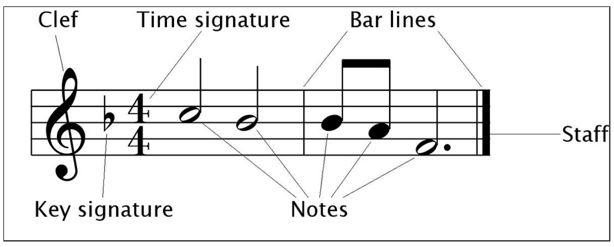


Figure 2.2: Five staff notation

Notes and rests indicate the duration of sounds and silences, with varying shapes of note-head or with the addition of note-stem and with beams or flags and positions on the staff determining their length. The pitch of a note is given by the position of its note-head on the staff and it can be modified by accidentals. Accidental is a symbol that indicates an alternation of a given pitch. The most common are the flat and the sharp, which represent an alteration of the tone by a half-tone. A pitch range of the staff is indicated by clef, which is usually written at the beginning of the staff. A clef is a musical symbol used to indicate which notes are represented by the lines and spaces on the musical staff. There are three kinds of clefs – the treble clef, bass clef and alto clef. After the clef can be written key signatures, which is a group of zero to seven sharp or flat signs on the staff to indicate which notes in the musical peace are flat or sharp. That is unless the aforementioned accidentals placed before certain notes say otherwise.

The key signature can be followed by a time signature. Time signature is a convention in music notation that specifies how many note values of a particular type are contained in each measure (also called bar). It consists of two numbers stacked vertically, such as 4/4, 3/4 or 12/8. The top number represents the number of beats in each measure. For example, in the 4/4 time signature, there are four beats per measure and in the 12/8 are 12 beats per measure. The bottom number in 4/4 indicates that those beats are quarter notes and in the 12/8 there are 12 beats per measure, with each beat being an eight note. Time signatures help musicians (or readers of the musical score) to interpret the rhythmic patterns of the music accurately. Changes in time signatures within a musical piece can add complexity to the musical piece, making it more interesting and vibrant.

Musical notation became a universal language of music. The principles of musical notation provide a common framework for musicians to interpret and perform music. It allows musicians from diverse backgrounds and places to communicate their ideas using a simple sheet of paper.

2.3.2 The first commercialized audio visualizer

The Atari Video Music is the first commercial electronic music visualizer. It was manufactured by Atari, Inc. in 1977. The system gets musical input from a Hi-Fi (high fidelity) stereo system and visualizes the musical intensity and mellowness into different shapes and colors and outputs them into the graphical display. It featured a set of knobs and switches that allowed users to manipulate the musical visualization in real-time, providing an interactive use. Two knobs were used for controlling the left and right audio input signals strength. These could increase the size of the visual pattern. One knob was for controlling colour. With this, you could increase the number of colours used in the visualization. The last two knobs were used to increase or decrease the geometric complexity of the shapes. The visuals produced by Atari Video Music were reminiscent of the psychedelic light shows, with swirling patterns and pulsating colours that responded to the music being played.

It was made to enhance the experience of listening to music, to make it more entertaining and relaxing. The concept was very innovative, but for the people of that time, it was relatively novel. Because of that, Atari Video Music was not commercially successful. Even so, it represented the first attempt to combine music and visuals, paving the way for future audio visualizations.

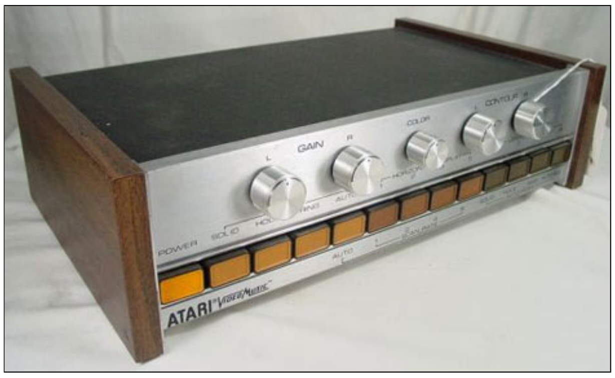


Figure 2.3: Atari video music [15]

2.3.3 A visualizer tool for deaf and hard of hearing

Audio visualizations are usually made to enhance the listening experience of musical pieces, but for the deaf and hard of hearing (DHH) community, the listening experience is greatly impoverished. But thanks to [12], the musical experience for the DHH community is greatly enhanced through the use of an on-screen visualizer generating effects alongside music.

Vitune is a visualizer tool that translates audio input into dynamic visualizations, allowing deaf individuals to perceive and engage with music through sight. The visualizer prototype is composed of two major modules: the analyzer module, and an actual visualizer module. The analyzer module analyzes the various components of music, including rhythm, pitch, and amplitude, and generates corresponding visual effects in real-time. The visualizer module takes the JSON [3] file containing the results of the analysis as input. These visualizations range from vibrant colour patterns and geometric shapes to animated graphics.

The impact of Vitune extends far beyond its technical capabilities, transcending the realm of assistive technology to become a catalyst for social inclusion and cultural empowerment within the deaf community. By providing deaf individuals with a means to experience music on their own terms, Vitune fosters a sense of connection, belonging, and participation in a world that is often dominated by auditory experiences. It empowers deaf individuals to engage with music in meaningful ways. Moreover, Vitune serves as a powerful tool for raising awareness and challenging misconceptions about deafness and sensory perception. By showcasing the inherent beauty and complexity of visual music.

Chapter 3

Related works

3.1 Multi view algorithm visualization

This [8] is about multi-view visualization and at the same time the biggest inspiration for our work. Brown created a software environment called BALSA (Brown ALgorithm Simulator and Animator), which mainly served for computer science education and researching and analysing of algorithms.

Balsa creates a real-time simulation of algorithms using high-resolution graphics and offers a dynamic graphic user interface for interaction with the algorithm visualization. It allows the user to create, delete and modify the size of the window, which shows the visualization to better suit the needs of the user.

Since it is multi-view, the user can create more than one window, each visualizing the algorithm differently. The first window can show the code that is currently being executed, for example, the merge-sort algorithm. It will show which line of the code is currently being executed. The next window can visualize the history of each iteration of the algorithm using horizontal lines of different lengths, each iteration having a separate column. In the third window, e.g. the merge-sort algorithm can be visualized by each element of the array being represented as a dot whose x coordinate corresponds to its position in the array, and whose y coordinate corresponds to its value. Each window can be resized and you can even select a window which you want to focus on. Balsa even saves the whole history of the visualization, so the user can go through it again, pause it or even reverse it.

Balsa is a great tool for studying algorithms from different perspectives. It allows us to more easily understand how different algorithms behave in various situations.

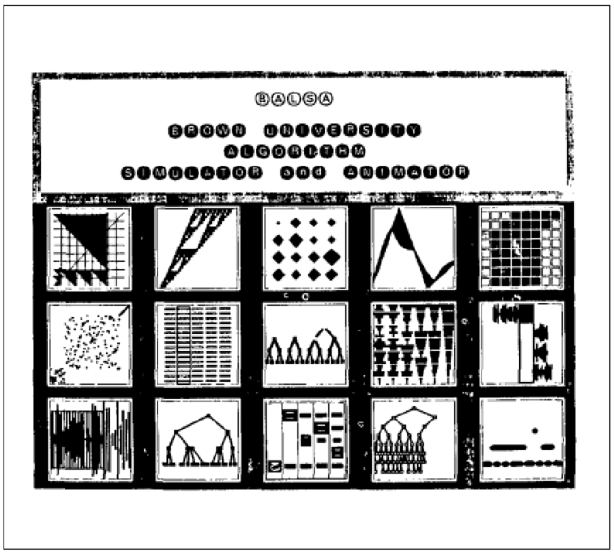


Figure 3.1: Balsa software screenshot [8]

3.2 Toroidal models

In music, we differentiate many perceptual models, such as timbre, pitch, keys and even emotions. For defining relationships between percepts, space is a natural medium. Spatial distances can be a representation of similarity between percepts. The nearer they are to each other, the more similar they are (the "better" relationship they have). In the music theory, the "Tonnetz" (German for "tone network") stands as a conceptual framework for understanding harmonic relationships, introduced by Leonhard Euler in the 18th century [16]. Tonnetz is a powerful tool for analyzing and visualizing musical structures, particularly in Western classical and contemporary music. It is represented as a two-dimensional lattice or a grid. It organises pitches based on specific musical relationships, such as perfect fifths, perfect fourths, and major thirds. In a typical Tonnetz, each point represents a musical note or pitch. The connections between points symbolize specific intervals or relationships. Horizontal lines represent perfect fifths. Vertical lines represent major thirds. Diagonal lines represent minor thirds. This layout allows for the visualization of chords, scales, and their transformations. Major and minor chords appear as triangles, while other harmonic structures can be perceived from different geometrical patterns.

Similar, but different to the Tonnetz is Weber's grid, named after music theorist Gottfried Weber, which is another framework used to describe and study musical relationships, exactly among chords and tonalities. He distinguished three kinds of triads (minor, major, diminished) and four kinds of seventh chords [16]. With regard to functional harmony, Weber emphasizes common tones between scales. Weber created a few charts describing tonal relationships, each being more extensive. In his more extensive chart of tonal relations, having 104 items, we can find multiple occurrences of the same tones.

Both Tonnetz and Weber's grid are usually visualized in 2D space, but they can be transformed into 3D. First, we cut out a strip from Weber's chart and rotated it to a horizontal orientation as is shown in Figure 3.2. In this strip, some keys and their enharmonic equivalents appear multiple times in different segments of the strip. To arrange a unique position for each key, we have to apply a spatial transformation by curling the strip into the shape of a tube in order to unite the minor keys of the upper row with those in the lower one (Figure 3.3). From there, we can join the duplicated keys at both ends of the tube to form a torus (donut shape) (Figure 3.4), which is still preserving the fifth relations and third relations.

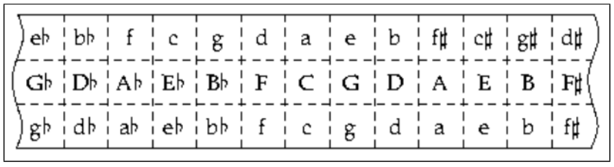

e♭ b♭ f c g d a e b f♯ c♯ g♯ d♯
G♭ D♭ A♭ E♭ B♭ F C G D A E B F♯
g♭ d♭ a♭ e♭ b♭ f c g d a e b f♯

Figure 3.2: Strip of keys cut out from Weber's chart of tone centres [16]

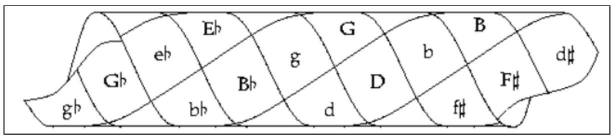


Figure 3.3: Strip of Weber's chart from Figure 3.2 curled into tube [16]

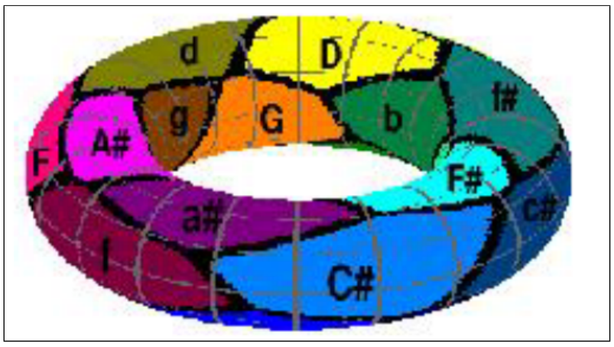


Figure 3.4: Both ends of the strip from Figure 3.3 are connected to form torus [16]

3.3 Spiral array model

The Chew's Spiral Array model [9] was developed by Elaine Chew, a professor of music and engineering. Unlike other models like Circle of Fifths or Tonnetz, which are only two-dimensional, Chew's Spiral Array model uses three dimensions to visualize the complexities of musical relationships. The Spiral Array is a geometric model that spatially represents pitches, chords (major and minor triads) and keys (major and minor) as points on the spiral configuration, as well as inside of the spiral, of the Harmonic Network.

The pitches, major chords, minor chords, major keys and minor keys are represented on five helices. The fundamental idea is to represent high-level objects in the spiral's interior as convex combinations of the representations of the lower components. The Spiral Array model begins with a formulation of the pitch helix, inner helices are generated as convex combinations of points on the outer ones. Each triad is represented as a point on the face of the triangle outlined by its component pitches. Each triad is a convex combination of its root, fifth, and third. The set of major triads forms a spiral inside the pitch spiral. Pitch classes belonging to a given key form compact clusters in the Spiral-Array model. The major key is represented by a spatial point in the interior of the three-dimensional spiral structures for the pitch classes and for the major triads. Like the major and minor triads, the major and minor keys also form spiral structures. The Spiral Array model is visualized using a set of spirals (Figure 3.5). Since the spiral model can represent objects from different hierarchical levels, the distance between the objects can be measured. This allows one to design computer programs to compute distances between objects and to make selections based on these distances.

The spiral Array Model offers a novel and comprehensive way to visualize and understand musical tonality, with a structure that accommodates a wide range of musical concepts and applications. By representing musical concepts through geometric structures, the model simplifies the analysis of complex musical relationships, making it useful for composers, music theorists, and educators.

The model and its real-time algorithms have been implemented in the tonal visualization software MuSA.RT [10], which has been used in music education videos and in live performances.

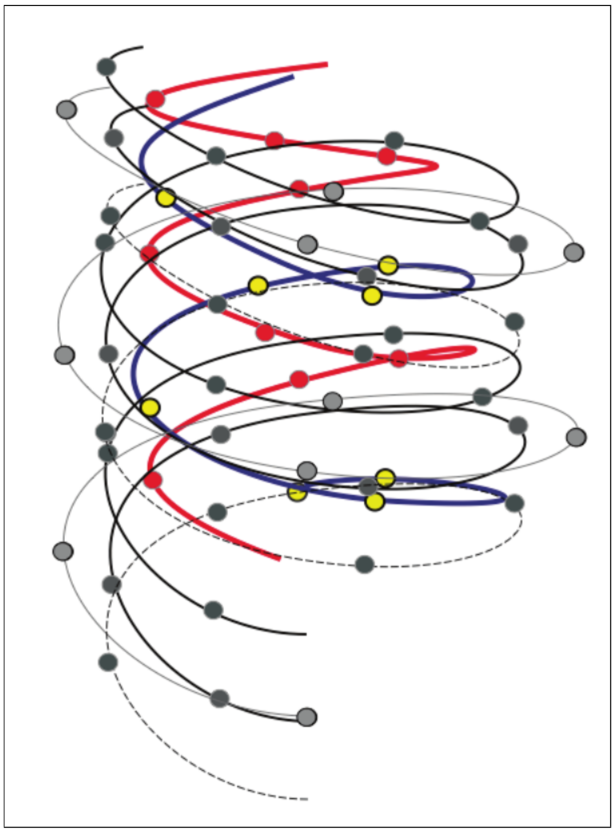


Figure 3.5: The Spiral Array model visualized as a set of 5 spirals [9]

3.4 Topology of music

Dmitri Tymoczko's article "Why Topology?" [25] explores the application of topological methods to music theory, focusing on voice leading and chord progression. He showcases how musical elements such as chords, scales, and voice leadings can be mapped onto topological spaces, displaying underlying patterns and symmetries. Topology is a mathematical discipline concerned with the properties of space that remain invariant under continuous transformations such as stretching and bending. In music, these properties can be used to model voice leading—how individual musical voices move from one chord to another.

Tymoczko uses higher-dimensional configuration spaces to model voice leading. These spaces, often visualized as twisted, mirrored orbifolds, can be harder to understand. Tymoczko simplifies these concepts by constructing two-dimensional diagrams that capture the essential features of these spaces. This approach helps to clarify how different voice-leading transformations can be visualized and analyzed within these spaces.

Tymoczko connects homotopy theory, which studies the properties of spaces that can be continuously deformed into each other, with transformational theory in music.

He shows how the fundamental group of voice-leading space can be used to record various kinds of loops and paths, representing different voice-leading transformations.

Tonnetz, which we mentioned earlier in this chapter, was extended by Tymoczko to arbitrary chords and set classes. He shows how this generalized Tonnetz can model voice leading among a wider variety of chords, providing a useful tool for analyzing complex chromatic passages.

Chapter 4

Project specification

4.1 Aim of the project

Visualizing music can encompass a wide range of elements, transforming auditory experiences into visual representations. We can visualize notes, pitch, tempo, musical harmony, chords, melody, tonality, keys, scales and even mood or emotions. Our project is mostly focused on visualizing notes and their metadata because we lack deeper knowledge of music theory to visualize more complex concepts like musical harmony. If we tried visualizing more complex musical elements, the visualization would have been probably unsuccessful.

We offer a few visualizations in a multi-view format similar to Brown [8]. Their implementations are presented in the "Implementation" sub-chapter.

4.2 Implementation

4.2.1 MIDI piano keys

First, we implemented the visualization of piano keys. MIDI can encode up to 128 different pitches, which is more than can be produced by a typical piano [5]. Piano usually has 88 keys and the shorter ones have 44 to 72 keys. Larger pianos have 97 keys and the largest have 108 keys. But since MIDI can encode up to 128 pitches, We modelled our MIDI keyboard to have 128 keys too, which is 10 and a little over half octaves [5].

The modelling was done in the software Blender [21]. Blender offers some basic pre-made meshes like plane, cube, circle, sphere and even a torus so we do not need to create a mesh object from scratch. These meshes can be accessed from the top panel by clicking on the "Add" button or by using a key shortcut "shift a" (this can be changed depending on the user's preferences). All keys were made by using these pre-made cube meshes from Blender, which we scaled and rotated to resemble the MIDI piano keys. Some of the white keys have black keys between them. To create this gap, we added another cube mesh and using the boolean modifier with a "difference" setting on, cut a section into the white key to create a space for the black key. All these keys used to make cuts into white keys are invisible to the camera and the user. All keys are identified by a unique number, ranging from 0 to 127 from left to right. This is because of the way we implemented the animation of the key being "pressed" in time when the key was being used in the musical piece.

The animation was done using a Blender add-on called Animation Nodes [18]. Animation Nodes is a node-based programming language. Node-based programming is a visual programming approach where the programmer instead of writing code connects nodes, each representing a different function or operation. Nodes are usually represented as squares or icons on a graphical interface and they are connected by lines or edges to indicate the flow of data and control between them. Node-based programming (or visual scripting) is usually used in game development, media and content creation or audio and music production.

First, we use the "Read MIDI file" node with a path to a MIDI file as an input. The output of this node is MIDI tracks from the chosen MIDI file, which is input for the next node - "Get list element". This node returns the element at the input index. This with the time info will be used as an input to the "Evaluate MIDI Track" node. The Animation nodes add-on already have a node implemented for reading a MIDI file, so we did not need to write a separate parser to it. From this reader, we can get a list of notes with their velocity being played. Since we are animating piano keys (with 128 keys) playing a song, we wanted to animate the keys being pressed (note on) and then released (note off). The reason we assigned different numbers to the keys, ranging from 0 to 127, was because the notes in the MIDI file are encoded as numbers. The leftmost note on the 128-key piano is C. In MIDI, this note is represented as 0, just like the identification name of our leftmost key. The next note is C# (C sharp) as 1, then D, D# as 2 and all the way to the last note G, 10 octaves higher than the first C, with the number 127. This way, the manipulation of the keys using just the number of the note becomes a lot more convenient. The key being pressed is visualized by rotating the key on its X-axis down, if the note is on, or back up to its original position if the note is off (releasing the key). Usually, the origin of the object is in the centre, so when we are rotating the object on its X-axis, the front side goes down, and the rear side goes up. Since we want the keys to mimic the motion of piano keys being pressed, we changed the origin of the keys to be at the back of the piano, so when the key is "pressed", only the front side goes down and the rear side stays relatively in the same spot end when the key is "released", it returns to its default position. The animation was done by the "Invoke subprogram" node. The input to this node was the list of notes and velocities we mentioned earlier and time (frame) info. From the list, all we need are the note values being played and time info. The note values are represented as numbers from 0 to 127, which we convert to text using the "Convert to text" node and give this string as input to the "Object by name" node, which allows us to manipulate with object with the same name as the given input. Then we use the "Combine Euler" node. This node composes an Euler from X, Y, and Z components [18]. We will only use the X information to rotate the selected (played) key object on the X-axis. But to rotate this object we need another node called "Object transforms output". This node sets the location, rotation, and scale of the input object to the input transformations [18]. In this node, we will only need to use a rotation. The input to this node is the object name from the "Object by name" node and X information from the "Combine Euler".

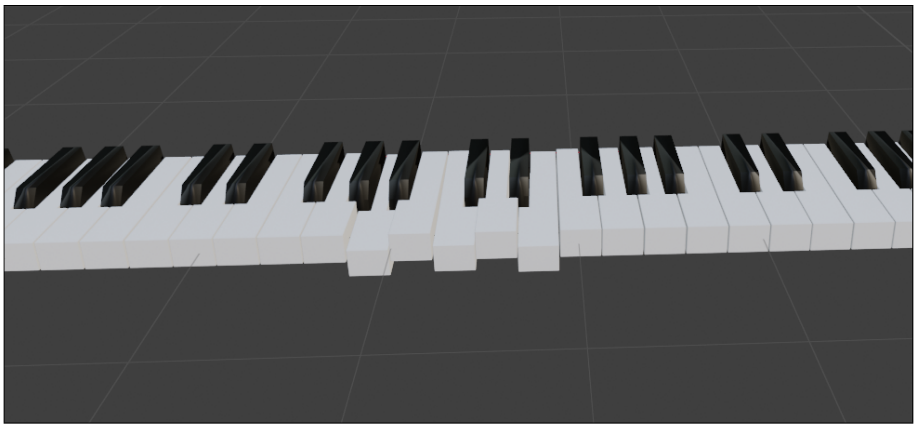


Figure 4.1: Visualization of MIDI keyboard with 128 keys

4.2.2 Notes

The second visualization's animation implementation was made similarly to the MIDI piano keys using the Animation Nodes. But this time the modelling was done differently. In the piano keys visualization we created a cube mesh from Blender meshes and then we scaled it manually without using commands and then duplicated each key and manually relocated it next to each other until we the whole octave and then we duplicated the whole octave 9 times. This method is pretty crude and can be inaccurate and tedious. This time to create our objects we wrote a simple script using Blender Python. When you first open Blender, you will find yourself in the "Layout" workspace window, which is used to create and manipulate mesh objects. Your current working window is shown in the top panel. To write a script in Blender, you need to change your working window to "Scripting". There we wrote a simple script that creates 128 spheres. The spheres represents different notes (C, C#, D, D#, E, F, F#, G, G#, A, A#, B). Each sphere has a different colour. The colours were chosen from [14].

First, we wanted to create only 12 spheres, one for each note, but we encountered a problem while implementing the animation. The first part of the implementation is the same as in the piano keys all the way until the "Invoke subprogram" node. There we wanted to use the "Math" node with a modulo of 12 function on the note number to get an integer from 0 to 12 representing all the notes on the octave. This would have been converted to text and used to manipulate the 12 sphere objects (each having a unique number as a name ranging from 0 to 11). But for some unknown reason, this did not work, so we were forced to create all 128 spheres representing each note. The animation is working on a similar principle as in the previous one. The only difference is that when the note is on, the object does not rotate on the X-axis but moves up, and when the note is off it moves down back to its original location. We use the same nodes but in the "Object transforms output" node, we do not use rotation but location and we move the spheres on the Z-axis. The camera is set so at first we do not see any sphere, only text representation of notes at the top of the screen. When the note is played, the sphere corresponding to that note will go up the Z-axis and stops right under the text representation of the played note. After that, the sphere returns down to it's starting position out of the camera's range.

The purpose of this visualization is to visualize which note or notes are being played in which time frame. This should better help the user analyse which notes are being played. However, differentiating between minor and major chords can be harder since MIDI does not differentiate between enharmonic notes.

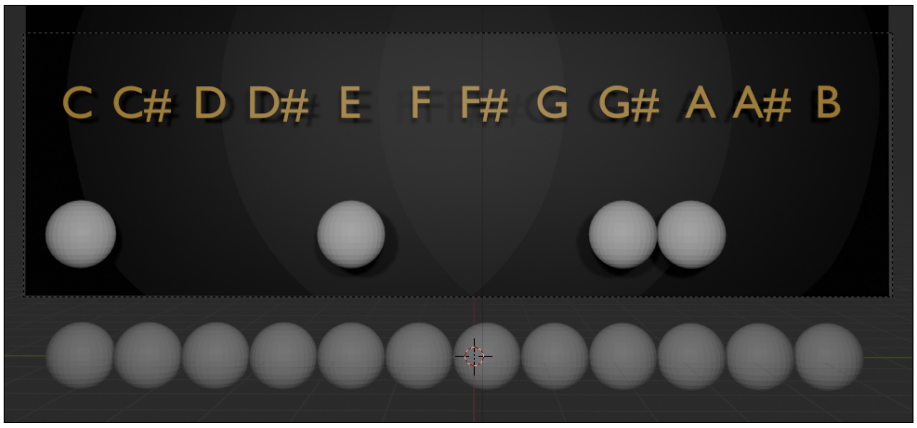


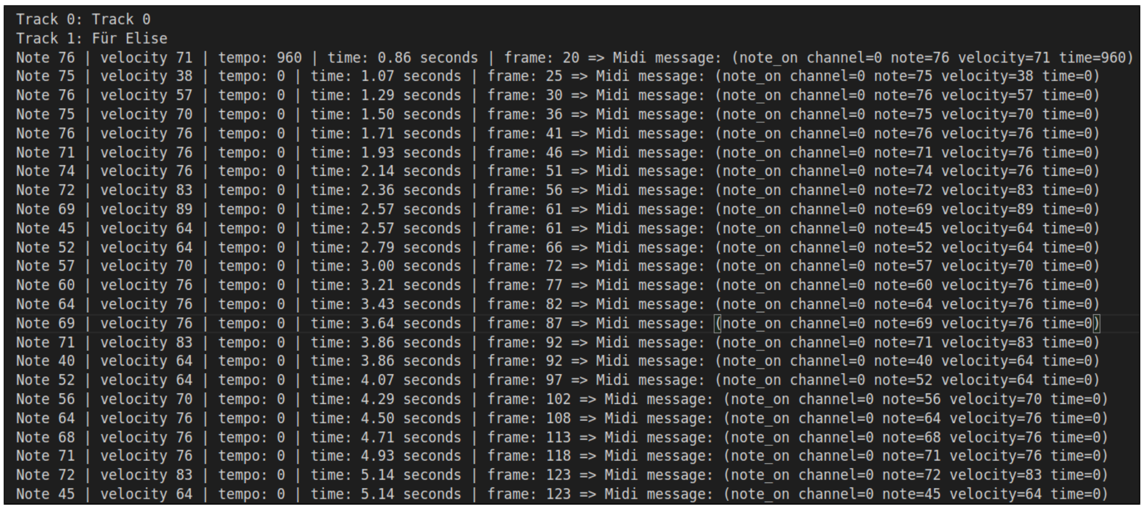
Figure 4.2: Visualization of notes as spheres.

4.2.3 MIDI parser

As we mentioned earlier, the input of our visualizations is a MIDI file. MIDI files contain all of the information about musical compositions, including notes, velocities, and timing information. This data is crucial for understanding the structure and performance of a piece of music. Visualizing MIDI data can transform abstract musical elements into concrete visual representations.

Occasionally, users may observe distinct events in the visualization but be uncertain about their underlying causes. To address this, we examine the MIDI file to identify the exact lines responsible for these specific events. Initially, we attempted to visualize this information in Blender using the 'Text' object, but this approach did not yield the desired results.

Consequently, we developed a Python script utilizing the 'Mido' [6] library to read MIDI files and created a parser for the inputted MIDI data. The script outputs a .txt file that organizes the musical information into a user-friendly format. This file includes the MIDI-encoded values for notes (ranging from 0 to 127) and the corresponding note names. Each line details the note's velocity and tempo. For easier analysis, we added time information in seconds for each note in the visualization, as well as frame numbers to facilitate better analysis within the Blender interface.



```
Track 0: Track 0
Track 1: Für Elise
Note 76 | velocity 71 | tempo: 960 | time: 0.86 seconds | frame: 20 => Midi message: (note_on channel=0 note=76 velocity=71 time=960)
Note 75 | velocity 38 | tempo: 0 | time: 1.07 seconds | frame: 25 => Midi message: (note_on channel=0 note=75 velocity=38 time=0)
Note 76 | velocity 57 | tempo: 0 | time: 1.29 seconds | frame: 30 => Midi message: (note_on channel=0 note=76 velocity=57 time=0)
Note 75 | velocity 70 | tempo: 0 | time: 1.50 seconds | frame: 36 => Midi message: (note_on channel=0 note=75 velocity=70 time=0)
Note 76 | velocity 76 | tempo: 0 | time: 1.71 seconds | frame: 41 => Midi message: (note_on channel=0 note=76 velocity=76 time=0)
Note 71 | velocity 76 | tempo: 0 | time: 1.93 seconds | frame: 46 => Midi message: (note_on channel=0 note=71 velocity=76 time=0)
Note 74 | velocity 76 | tempo: 0 | time: 2.14 seconds | frame: 51 => Midi message: (note_on channel=0 note=74 velocity=76 time=0)
Note 72 | velocity 83 | tempo: 0 | time: 2.36 seconds | frame: 56 => Midi message: (note_on channel=0 note=72 velocity=83 time=0)
Note 69 | velocity 89 | tempo: 0 | time: 2.57 seconds | frame: 61 => Midi message: (note_on channel=0 note=69 velocity=89 time=0)
Note 45 | velocity 64 | tempo: 0 | time: 2.57 seconds | frame: 61 => Midi message: (note_on channel=0 note=45 velocity=64 time=0)
Note 52 | velocity 64 | tempo: 0 | time: 2.79 seconds | frame: 66 => Midi message: (note_on channel=0 note=52 velocity=64 time=0)
Note 57 | velocity 70 | tempo: 0 | time: 3.00 seconds | frame: 72 => Midi message: (note_on channel=0 note=57 velocity=70 time=0)
Note 60 | velocity 76 | tempo: 0 | time: 3.21 seconds | frame: 77 => Midi message: (note_on channel=0 note=60 velocity=76 time=0)
Note 64 | velocity 76 | tempo: 0 | time: 3.43 seconds | frame: 82 => Midi message: (note_on channel=0 note=64 velocity=76 time=0)
Note 69 | velocity 76 | tempo: 0 | time: 3.64 seconds | frame: 87 => Midi message: (note_on channel=0 note=69 velocity=76 time=0)
Note 71 | velocity 83 | tempo: 0 | time: 3.86 seconds | frame: 92 => Midi message: (note_on channel=0 note=71 velocity=83 time=0)
Note 40 | velocity 64 | tempo: 0 | time: 3.86 seconds | frame: 92 => Midi message: (note_on channel=0 note=40 velocity=64 time=0)
Note 52 | velocity 64 | tempo: 0 | time: 4.07 seconds | frame: 97 => Midi message: (note_on channel=0 note=52 velocity=64 time=0)
Note 56 | velocity 70 | tempo: 0 | time: 4.29 seconds | frame: 102 => Midi message: (note_on channel=0 note=56 velocity=70 time=0)
Note 64 | velocity 76 | tempo: 0 | time: 4.50 seconds | frame: 108 => Midi message: (note_on channel=0 note=64 velocity=76 time=0)
Note 68 | velocity 76 | tempo: 0 | time: 4.71 seconds | frame: 113 => Midi message: (note_on channel=0 note=68 velocity=76 time=0)
Note 71 | velocity 76 | tempo: 0 | time: 4.93 seconds | frame: 118 => Midi message: (note_on channel=0 note=71 velocity=76 time=0)
Note 72 | velocity 83 | tempo: 0 | time: 5.14 seconds | frame: 123 => Midi message: (note_on channel=0 note=72 velocity=83 time=0)
Note 45 | velocity 64 | tempo: 0 | time: 5.14 seconds | frame: 123 => Midi message: (note_on channel=0 note=45 velocity=64 time=0)
```

Figure 4.3: Parsed MIDI File

4.2.4 2D toroid

In Chapter 3 we mentioned how can be music visualized using Toroidal models. Toroid (Figure 3.4) is a 3D object resembling a donut. This visualization visualizes key relationships. Blender offers toroid mesh in the mesh menu, but the remodelling of its faces to match the intricate position of each key visualization is beyond my scope of skill furthermore, to capture all the events that could happen, we would need to render each visualization from 4 different views, which would be too time-consuming.

That's why we decided to model its 2D variant from [16]. To model this, we didn't use any of the meshes Blender offers. We created our own mesh. First, we needed to define all vertex positions for each key. Each key is made of five vertexes, which creates one face of the mesh. To create separate mesh data for each key, we first needed to "register" it using "bpy.data.meshes.new(name)" function. Then we can create a new mesh using "from_pydata(verts, edges, faces)". Arguments to this function are list of five vertexes, a list of edges (this list can be empty) and list of faces (sequence of indexes to the vertex list). Then we can create an object using "bpy.data.objects.new()" function. We will repeat this another 23 times. Then we can run this script and it will create our desired two-dimensional representation of key relationships (Figure 4.4). Then we can create a material (the term for colour in blender) for each key (face/ key object). This can be done manually by selecting each face and assigning it a colour or in script.

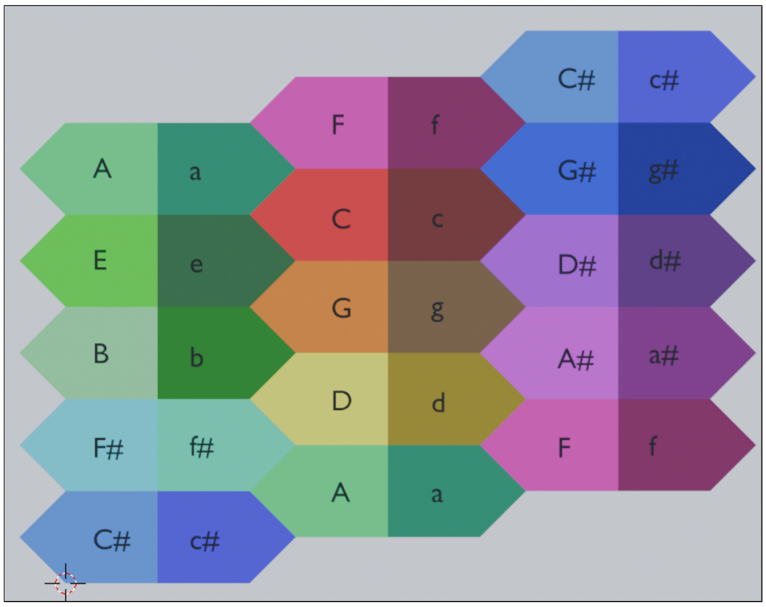


Figure 4.4: 2D representation of key relationships

This visualization can't be done using Animation Nodes. Firstly, we would need to analyze the whole MIDI file and recognize all enharmonic notes. For this we can write a special parser or find an already implemented one. The animation would show a cursor travelling above the grid (Figure 4.4) depending on the key played. This will be implemented in the future.

Chapter 5

Problems and future work

5.1 Problems

During the implementation phase of this work we encountered some unpleasant problems and inconveniences. The first inconvenience was our equipment. Rendering in Blender and probably in any other animating software is very demanding on computational power. A simple ten-second animation of piano playing music took more than half an hour to render. To render the animations, we used EEVEE rendering engine, which is used in game development for real-time rendering because of its fast speed.

Another inconvenience was with rendering animation with audio. We are visualizing music based on the information in the MIDI file. This file contains information about music, like when which for how long and how high is note being played. And the problem was, that Blender does not offer to render animation by using MIDI file audio. Blender only offers MP3, FLAC and a few other encoding standards. This caused some complications because we needed to find an mp3 encoding of our MIDI file audio and the problem was caused by different starting times of our songs. The mp3 file started playing the musical piece a few seconds later than the MIDI file, so we needed to manually fit the start of the mp3 file into frames to match the visualization. Or the mp3 audio can be slightly faster than the MIDI. This will cause desynchronization between audio and visualisation.

Another problem is with the MIDI file itself. Music contains notes that are enharmonic. Enharmonic notes refer to two or more notes that sound the same in pitch but are written or notated differently. This problem causes the structure of the Western musical scale and the use of sharps and flats to indicate different tonalities. In the context of a keyboard instrument like the piano, each key represents a specific pitch. Some of these pitches can be represented by more than one note name, depending on the musical context or key signature. For example, the pitch C# (C sharp) can also be referred to as Db (D flat). Though they are named differently, they are played on the same key on a piano and produce the same sound. The another enharmonic notes are: C# = Db, D# = Eb, F# = Gb, G# = Ab, A# = Bb. MIDI file does not differentiate between enharmonic notes. MIDI only encodes both enharmonic notes as the note with the "#" signature. That complicates things when we want to visualize music. Lot of music visualizations mentioned in Chapter Three differentiate between enharmonic keys, minor and major chords and minor and major keys. These all depend on enharmonic notes. If we do not differentiate between them, the visualization can become inaccurate or even impossible to made. But that does not mean we can't find out which enharmonic note is being played. As mentioned earlier, it depends on the context of music. This can be analyzed by some analyzing programs, like for example Šukola's Software for Analysis of Harmony in Music [13].

The next problem was really sudden and unexpected. For our music visualization, we used a Blender add-on called Animation Nodes. For manipulating different objects we used the "Object by name" node. This node allowed us to manipulate different objects by giving them their names as input. This node made our work easier and more intuitive. That was until the node suddenly disappeared from our workspace and we couldn't find it. The only way to use it again was to make a copy of the already made visualization which used this node and than delete everything except that node.

5.2 Further ideas

The next goal of this work will be inspired much more by Brown's BALSA software [8]. Right now, we only offer a few ways to visualize a musical piece, but we do not offer the interactive user interface like BALSA does. The reason for this is that the whole project was made in Blender, which is mainly used for creating animations, modelling and sculpting unlike game engines like Unity or Unreal Engine which can even provide with user interface.

Initially, we wanted to create every visualization together in a single project, but than we found out the realization would be unrealistic. The first problem would be with our lack of computational power. Another problem would be with a camera. The camera would need to be too far away from the scene to capture everything and lots of details would not be visible. The solution was to model and animate each visualization separately, but this prevented our goal of multi-view visualization. To achieve our goal of multi-view visualization, we had to render every visualization animation separately and later put them together into a single video. But this approach is very time-consuming, so for our future work we hope to find a way to make our visualization interactive. The next step will be to find a way to integrate these visualizations to another software or game engine to create an interactive user interface. We would like it to be relatively easy, that even a child would find it intuitive to use. There will be a few buttons to select which visualizations the user wants to see and another to start the visualizations. When the visualization starts the user would see all the visualizations that he selected on different windows, which would allow the user to select one of them to focus on if he saw something interesting. The user interface would allow the user to pause the visualization and even revers it back. Depending on the computational power of the user's equipment, rendering visualization can take an astronomically long time.

Another feature which we want to add is song selection. If the user wants to visualize a song of his choice, he would need to access the source code of every visualization and manually change the file path to the song he wants. We want to automate it. One possibility would be to create a button in the user interface which would allow the user to import his own MIDI file or even offer a selection of available MIDI files. With this, we will need to implement a function that will automatically adjust the length of the animation according to the length of the song. This needs to be implemented in Blender. Information about the song length can be found in the MIDI parser (Chapter 4.2.3)

Some of the visualizations are three-dimensional. In more advanced or complex visualizations this could pose a problem that most people don't even think about. When you are looking at a three-dimensional object, you can see it only from the front (or where the camera is currently looking), but not the rear. This could be a problem in visualizations focusing on musical analysis because we can miss some useful data just because we couldn't see it. We could give the user the ability to freely move with the camera in the visualization, but there would always be a part of the visualization that we couldn't see. A possible solution would be to allow the user to set a secondary camera focusing on the rear of the visualization and even allow the user to split the current window diagonally or horizontally.

We started this project without any knowledge of music theory, so our start was harder than we expected. Our next plan will be to dive deeper into music theory so we can create more complex music visualizations. If we tried to implement some of the complex music visualizations (e.g. The Spiral model), the result would be unsatisfactory and most likely even inaccurate, which could lead to incorrect music analysis, and that is what we want to avoid.

Another idea was given to me during one of the discussions, and that is for the user to create his/her visualization.

Conclusion

In this thesis, we explored various methods for visualizing music, focusing on multi-view visualization inspired by Brown's work. While our initial goal was to develop a comprehensive multi-view visualization software for classical music, we faced significant limitations due to the computational power of our equipment. The complexity of handling and animating numerous objects in Blender exceeded our capabilities, necessitating adjustments to our original plan.

Despite these challenges, we successfully integrated selected visualizations and original techniques to create a multi-view representation of music. This project serves as a foundation for future enhancements and hopefully offers valuable insights for musicians and enthusiasts interested in the visual aspects of music. Looking ahead, further improvements in computational resources and software optimization could enable the realization of more sophisticated and interactive visualizations, expanding the potential applications of this work.

References

[1] Advancements after world war ii. https://www.britannica.com/topic/music-recording/Advancements-after-World-War-II.

[2] The complete midi 1.0 detailed specification. http://www.shclemen.com/download/The%20Complete%20MIDI1.0%20Detailed%20Spec.pdf.

[3] Introducing json. https://www.json.org/json-en.html.

[4] Renaissance music. https://www.vam.ac.uk/articles/renaissance-music.

[5] Standard midi file format. https://faydoc.tripod.com/formats/mid.htm.

[6] *MIDI Objects for Python*, 2023.

[7] Britannica. Evolution of western staff notation. https://www.britannica.com/art/musical-notation/Evolution-of-Western-staff-notation.

[8] Marc H. Brown and Robert Sedgewick. Techniques for algorithm animation. *IEEE Softw.*, 2(1):28–39, 1985.

[9] Elaine Chew. The spiral array: An algorithm for determining key boundaries. In *International Conference on Music and Artificial Intelligence*, pages 18–31. Springer, 2002.

[10] Elaine Chew and Alexandre R.J. Francois. Musa.rt: music on the spiral array. real-time. In *Proceedings of the Eleventh ACM International Conference on Multimedia*, MULTIMEDIA '03, page 448–449, New York, NY, USA, 2003. Association for Computing Machinery.

[11] Richard E. Cytowic. *Synesthesia: A Union of the Senses (Second Edition)*. The MIT Press, 2002.

[12] Jordan Aiko Deja, Alexczar Dela Torre, Hans Joshua Lee, Jose Florencio IV Ciriaco, and Carlo Miguel Eroles. *ViTune: A Visualizer Tool to Allow the Deaf and Hard of Hearing to See Music With Their Eyes*. ACM, 2020.

[13] Michal Šukola Eva Ferková. Software for analysis of harmony in music. https://dev.analysisofharmony.sk/, 2020.

[14] Flutopedia. The color of sound. http://www.flutopedia.com/sound_color.htm.

[15] James Grahame. Atari video music: Psychedelic 1970s music visualizer. https://www.retrothing.com/2007/09/atari-video-mus.html.

[16] K Obermayer H Purwins, B Blankertz. Computing in musicology. https://www.researchgate.net/profile/Hendrik-Purwins/publication/266882670_5_Toroidal_Models_in_Tonal_Theory_and_Pitch-_Class_Analysis/links/550f22b70cf2752610a00b22/5-Toroidal-Models-in-Tonal-Theory-and-Pitch-Class-Analysis.pdf, 2007. online.

[17] Colette Hemingway and Seán Hemingway. Music in ancient greece. https://www.metmuseum.org/toah/hd/grmu/hd_grmu.htm, 2001.

[18] Jacques Lucke. Animation nodes. https://github.com/JacquesLucke/animation_nodes.

[19] Steven Brown Nils L. Wallin, Bjorn Merker. *The Origins of Music*. MIT Press, 2001.

[20] The Editors of Encyclopaedia. "gregorian chant". https://www.britannica.com/art/Gregorian-chant, 2024. "accessed 20.4. 2024".

[21] Ton Roosendaal. Blender. https://www.blender.org/.

[22] Joseph Rothstein. *MIDI: a comprehensive introduction, second edition*. A-R Editions, Inc., 1995.

[23] Andrew Sears and Julie A. Jacko. *Human-Computer Interaction: Design Issues, Solutions, and Applications*. CRC Press, 2009.

[24] Randy Skretvedt and Christopher H. Sterling. radio. https://www.britannica.com/topic/radio.

[25] Dmitri Tymoczko. *Journal of Mathematics and Music Mathematical and Computational Approaches to Music Theory, Analysis, Composition and Performance*. Journal of Mathematics and Music, 2020.

[26] Šimon Prokop. Visualization of musical harmony. http://scripting.molab.eu/tutorials/visualization-of-musical-harmony/, 2012.

Appendix A: Digital material

All implementations of musical visualizations mentioned in the thesis, along with other supplementary materials can be found in the electronic attachment. The latest version of our work can be found here:
`https://github.com/Maax02/Multi-view-visualization/tree/main`