Università degli Studi di Padova

DIPARTIMENTO
DI INGEGNERIA
DELL'INFORMAZIONE

DEPARTMENT OF INFORMATION ENGINEERING
Master Degree in Ict for Internet and Multimedia

A New Sunrise for Speech Therapy: Development of SoundRise 2.0 Application

STUDENT: Giada Zuccolo, ID. 2055702

SUPERVISOR: Professor Sergio Canazza Targon

CO-SUPERVISOR: Dott. Alessandro Fiordelmondo

Academic Year 2022/2023
December 14, 2023

Zuccolo Giada: *A New Sunrise for Speech Therapy: Development of SoundRise 2.0 Application*, Master's Thesis, © December 2023

To those who believe in the beauty of their dreams

Ringraziamenti

Alla mia famiglia. Alla mia mamma e al mio papà, Michela e Alessandro. Il mio porto e posto sicuro. Il mio sostegno e la mia forza. Grazie per ogni cosa. Ai miei fratellini, Riccardo e Alice. Grazie per i momenti di leggerezza che inconsapevolmente mi regalate, per tutte le coccole e per essere semplicemente voi.

Ai miei nonni, Bruno e Graziella, Giovanni e Giannina. Siete, e sarete per sempre, presenza costante nel mio cuore. Grazie per essere stati al mio fianco, con belle parole e tante carezze.

A tutti i miei parenti. Grazie perchè mi avete fatto crescere sentendomi amata e per non avermi mai fatto mancare tante parole di affetto, stima e incoraggiamento. A Valentina, parente aggiunta, grazie per essere una spalla di fiducia, sempre presente.

A tutti i miei amici: voi non lo sapete davvero il bene che mi fate, con una risata, una serata in compagnia o semplicemente essendoci. In particolare, grazie Martina per essere sempre stata al mio fianco nel mio periodo difficile e per avermi aiutata a trovare la mia serenità. A tutti quei momenti lì, che ci hanno fatto stare bene. Un enorme grazie a Sofia: dai banchi della triennale ne sono passate di avventure, e la cosa bella è stata sentirti vicina, a viverle tutte insieme anche da distanti, diventando sempre più amiche, anche se le nostre strade universitarie si sono divise. Grazie di cuore Arina e al caso che ci ha fatto incontrare. Un grazie infinite per il supporto e la presenza che solo tu hai saputo darmi in questi due anni di magistrale, e grazie delle belle parole di affetto che mi hai sempre dedicato.

Alla musica, battito che da sempre anima il mio cuore. Grazie per tutte le emozioni che mi fai vivere e per le tante persone che mi hai fatto conoscere. In particolare, grazie Elena, perchè sei entrata nella mia vita in punta dei piedi ma battendo con energia la tua batteria, e sei diventata per me una delle persone più importanti con la quale posso confidarmi, sapendo di poter essere davvero me stessa. Un grazie speciale a Giulia, per me non sei solo la mia (bravissima) insegnante di canto. Grazie per la bella anima che sei, per ascoltare non solo le note che canto ma anche le emozioni che sento e per abbracciarle con il tuo grande cuore.

Grazie a tutti, e anche a quelli che non ho citato direttamente. Tutti voi sarete sempre nella mia tasca a destra in alto.

Infine, un importante grazie al prof. Sergio Canazza e al dott. Alessandro Fiordelmondo, per avermi dato l'opportunità di sviluppare un progetto di tesi che fin dall'inizio mi è stato molto a cuore. Vi ringrazio per la stima e fiducia che, anche umanamente, mi avete dimostrato fin da subito, e per tutto il sostegno che non mi avete mai fatto mancare.

Abstract

SoundRise 2.0 is an application developed in React JS with the aid of Web Audio API, designed to help people with communication difficulties, especially children. The application provides an intuitive interface that allows the user to practise speech therapy voice exercises independently. The interface consists of a sun that symbolises the user's voice and is animated according to the tonal and timbral characteristics of the voice itself: it analyses these vocal characteristics and displays a correspondent visual feedback in real time.

This application is the result of years of study and research by the CSC (Centro di Sonologia Computazionale). For years, the CSC has aimed to promote learning activities that encourage children, by considering physical actions as an integral part of cognition, through the use of technological enhancement.
During these years, a number of prototypes have been developed and then shelved. The aim of this thesis is to develop an application maintainable over time, undertaking a reactivation process previously designed at CSC for the preservation and reactivation of artefacts.

The thesis gives an overview of the state of the art and defines the reactivation process that has to be carried out.
All the technical aspects of audio data analysis will be explained and examined, in addition to future SoundRise 2.0 enhancements, in order to improve users' communication capabilities and overall quality of life.

Contents

List of Figures

1 Introduction

Soundrise is an application designed to help people with communication difficulties, especially children. The main idea is to provide an intuitive and inclusive interface that allows the user to practice speech therapy vocal exercises independently.

The protagonist of this game is a sun which symbolizes the user's voice. Through its position on the horizon, its size, and its color, it provides graphic feedback consistent with the vocal characteristics received from the microphone: its position in the vertical axis of the sun to the horizon corresponds to the pitch, while its size corresponds to the intensity of the sound. The duration of the sound is represented by the smiling face of the sun opening or closing its eyes. Furthermore, the sun can be colored with 5 different colors with which the five vowels of the Italian language are represented. The colors to match the vowels were chosen based on a study on the non-random associations between graphemes and colors in synesthetic and non-synesthetic populations: it was decided to represent *a* with *red*, *o* with *orange*, *e* with *green*, *i* with *blue*, and *u* with *gray*.

The first idea for SoundRise was sketched out by Prof. Federico Avanzini, Prof. Antonio Rodà and Prof. Sergio Canazza from the Department of Information Engineering at the University of Padua, in collaboration with Dr. Serena Zanolla from the University of Udine. From then on, the project started to develop in the CSC (Centro di Sonologia Computazionale) at the University of Padua.
In the CSC, the study of the preservation and reactivation of artefacts does not only relate to the artwork. The aim is to have a reactivation and preservation process that

is valid for several areas, including the development (and maintenance) of applications. This is the case of SoundRise application: the intended goal was to perform *reactivation operations* to give this application the longevity and sustainability it deserves.

In the field of teaching aimed at children, tools based on multimedia technologies can be used together with traditional teaching methods. SoundRise is therefore designed as an interactive multimodal application for teaching, proposed as a new way of teaching children the characteristics of their own voice through an exploration of vocal abilities, ensuring that the child himself discovers, and then interprets and understands, the information content it receives through graphic feedback. The child plays an active role, as he controls the graphic evolution of the feedback through his own voice. This offers an incentive to use the application, with the possibility of obtaining greater commitment and time dedicated to using SoundRise.

1.1 Document Structure

A detailed subdivision and preview of the chapters is now described.

In the second chapter an overview of speech therapy treatment for supporting the growth of deaf children is initially presented. Following that, the importance of customized and targeted education is explained, highlighting how the amalgamation of technology and gaming contributes to a successful solution, forming the basis of the SoundRise concept.

The third chapter explains the basic idea and general functioning of the SoundRise application, describing in the deatils the past versions of this application. In particular, a focus will be placed on its reactivation according to models studied and defined within the CSC, and the reactivation process applied to SoundRise will be outlined.

In the fourth chapter, following the provide of a theoretical foundation regarding sound and the relevant characteristics for the SoundRise application, a detailed outline of the technologies employed in the development of SoundRise will be presented and analyzed, elucidating the reasons why they were chosen.

The fifth chapter delineates the functionality of the SoundRise application, elucidating various scenarios in which the application operates. It highlights the generation of corresponding mechanisms, the consequential functions, and the interface modifications specific to each scenario.

In the sixth chapter, the strings of the project are pulled, analyzing the result reactivation process of the SoundRise application. Furthermore, some tips about future improvments and developments are indicated.

All the various references are reported at the end of the document.

2 A dive into the world of children's speech therapy

In this chapter, an overview of speech therapy treatment to support the growth of deaf children is initially presented. Then, focusing on the importance of customised and targeted education, it will be explained how the combination of technology and gaming leads to a winning solution which is the basis of the idea of SoundRise.

2.1 A Necessary Premise

Don't say "the deaf" - use "Deaf people". Also avoid judgemental phrases such as: "suffering from deafness", "afflicted by deafness" or "trapped in a world of silence".
- British Deaf Association, Definitions of hearing impairments [1]

The appropriate terminology to refer to deaf people is *deaf*. This term is used to refer to people who self-identify as deaf, deaf-blind, deaf-disabled, hard of hearing, late-deafened, hard of hearing, and hearing impaired. Lowercase *deaf* is used to refer to the audiological condition of hearing loss, while uppercase *Deaf* is used to refer to the socio-cultural community perspective and not from a medical perspective.
This is an important differentiation to the Deaf community, since the medical model focuses on curing the disease, and the social model focuses on curing the society that

creates unnecessary barriers [2].

There are some terms that are mistakenly used in reference to deaf people, such as *deaf-mute*, *deaf-and-dumb*, and *deaf and silent*. They are considered offensive to the community and also reveal a certain backwardness of mind [3].

Deaf-mute is a term that historically was used to identify people who were deaf and, for this reason, unable to communicate. Hence, used to name all those people who, in addition to their deafness, also cannot speak an oral language or have a certain degree of ability to speak, but opt not to speak because of the negative or unwanted attention that atypical voices sometimes attract. This is bad and especially incorrect, because various methods of communication can be used even without the voice (e.g. sign language): a communication occurs when one's message is understood by others and they can respond.

The term *deaf and dumb* was uttered by Aristotle, who considered deaf people as incapable of learning and reasoned thoughts. According to his thinking, if someone could not use their own voice, that person had no chance of acquiring any cognitive skills. In addition to this, *dumb*, not in the sense of *mute*, has a second meaning: *stupid*. Therefore, using the term *dumb* also implies thinking that such individuals are stupid, which is obviously not the case [4].

Indeed, throughout history, many deaf and hard-of-hearing individuals have made significant contributions to society through inventions and life experiences.
Thomas Edison, despite scarlet fever affecting his hearing, was a key inventor. *Ludwig van Beethoven*, a renowned classical composer, continued to create masterpieces despite his deafness. *Alexander Graham Bell*, whose mother was almost totally deaf, dedicated his life to deaf education, conducting groundbreaking communication and sound trasmission experiments. *Terence Parkin*, a deaf South African swimmer, has won numerous Olympic and World Championship medals, proving that people with disabilities can excel in sports.

All these extraordinary life stories, together with many others not mentioned, pro-

vide a testimony of the spirit of people who live with disabilities with courage, inspiring and reminding us that human potential is boundless, and can transcend the barriers imposed by circumstances.

This is the reason why it is still important to work on the improvement of the lives of people with disabilities, helping them to improve their situation and to compensate for their difficulties: the ongoing research of innovative solutions and applications by researchers, doctors and scientists are inspired by the profound conviction that every individual, regardless of the challenges they face, possesses extraordinary potential that must not remain blocked.

2.2 Speech therapy notions

Speech therapy is a paramedical profession that deals with the diagnosis, prevention, evaluation and treatment of communication, language and voice disorders and related cognitive disorders (e.g. related to memory and learning).
In the **developmental age**, speech therapy deals with all activities for the rehabilitation of children with speech and language disorders. It teaches them to communicate in an effective way and to express emotions, needs and thoughts, ensuring that they do not feel isolated or unable to communicate with others. It plays a crucial role in their childhood process because it helps these children to acquire language skills through gestural communication, the use of assistive communication devices, and **sign language**.

Italian Sign Language (LIS)[5] is a visual and gestural language used by the Italian deaf community as the primary means of communication. It is independent of spoken and written Italian language and is the mother tongue of many deaf people in Italy. LIS is complex and articulate, consisting of hand signs, facial expressions, body movements, and gestures that express ideas, emotions, and concepts. It is used to form sentences and conversations, similar to spoken language.

LIS is different from sign languages used globally: each country has its own sign language, each with its own linguistic and grammatical characteristics.

Speech therapy supports the integration of deaf children in schools, working with teachers to develop teaching strategies and communication methods adapted to each child's needs, ensuring that they have the same educational opportunities as their hearing peers.

2.2.1 A brief history of speech therapy

Speech therapy as a discipline intimately connected to communication finds its deepest roots in ancient civilisations and those of the East. From the Assyrians to the Babylonians, from the Sumerians to the Greeks, from Indian medicine to Chinese and Arabic medicine. One of the earliest "therapies" was that of Demosthenes, who used pebbles to help correct a speech defect [6].
The roots of speech therapy in Europe date back to the medieval period, when the first attempts to treat speech disorders were often based on superstitious beliefs or religious practices. However, the real evolution of speech therapy began in the 19th century, when scientific attention shifted to the study of language and its disorders. In the 19th century, professionals from different fields in Europe dealt with alterations in human communication [7]. In the early years of the 20th century, speech therapists worked mainly in the field of speech delays and speech defects. Very soon, however, they also began to deal with voice problems and hearing impaired individuals, as well as speech disorders (such as aphasia and dyslexia) and stuttering. The profession of speech therapist is therefore relatively young and developed mainly during the 20th century [8].

In Italy, the development of speech therapy has been influenced by cultural transformations and organizational changes within the health system. During the Italian Unification (19th century), health and rehabilitation were considered private spheres, and speech therapy was not established. The profession emerged from spe-

cializations in teachers, primarily focusing on the school system. Historically, speech therapy in Italy had its origins in those who dealt with voice production and correct articulation, including singing, diction, deaf, and disabled educators. The first schools for speech therapists were opened in the late 1960s, with the profession becoming increasingly health-focused. The profile of speech therapists evolved parallel to healthcare and the concept of health [9].

In recent decades, speech therapy has continued to evolve. The education and training of speech therapists have become increasingly specialised, and the use of technology has made new diagnostic and therapeutic tools available. In addition, the focus has broadened to include a wider range of language disorders, including those related to autism and neuroscience.

At the European level, the European Union has played an important role in promoting best practices in speech therapy through knowledge sharing and harmonisation of professional standards. This has contributed to improving the quality of speech therapy services across the continent.

2.2.2 Approaches and methods

One of the main goals of speech therapy is to facilitate effective communication. This often involves improving lip-reading skills, improving speech articulation and helping people to better understand and interpret spoken language. Deaf people can receive speech therapy early in life, in particular if they use hearing aids or cochlear implants. Therapy is customised to help them in developing speech communication skills and integrating them into their day-to-day life.

The speech therapist provides specific exercises designed to reinforce listening awareness, auditory detection, and discrimination skills, and then targeted to support the development of early communication and speech skills. Each individual has specific goals and a customized treatment plan redicted by the speech therapist. Therapy

sessions often include exercises and activities designed to improve speech clarification and articulation. Such techniques as practicing vowel and consonant sounds, as sentence pronunciation, and learning to control intonation and pitch are common components of speech therapy [10].

A widely used method for treating these cases is called Auditory Verbal Therapy (**AVT**) [11]. It is oriented to the optimal acquisition of verbal language through listening, thus through the enhancement of the auditory and verbal channel. This is possible through the creation of a suitable setting and the use of auditory stimulation with music, songs, sounds and many other activities. The AVT method emphasises the recognition and interpretation of sounds and spoken language, encouraging children to develop active and passive listening skills. In addition to the speech therapist, the child's family plays a key role in AVT. Parents are trained to help children develop language and communication skills in everyday contexts.

An alternative approach is called the **bimodal method**. The term *bimodal* refers to the use of two primary modes of communication at the same time: spoken language and visual-gestural communication through sign language [12].
The bimodal method aims to provide children with a wider range of communication skills because it focuses on helping deaf children to understand the language rather than to produce it. Sign language can be used to develop a solid basis for visual and gestural communication, while spoken language can be taught to help children interact with the auditory world and communicate with hearing people. The bimodal method is often used when there are different degrees of deafness and when the child's family wishes to offer the child the possibility of using both communication modes. The choice to use the bimodal method is a personal decision and varies from family to family, depending on the needs and preferences of the child and his/her family. In general, the aim of the bimodal method is to provide deaf children with a wider range of communicative tools, enabling them to interact effectively with both hearing and deaf people, thus contributing to their social and communicative development.

2.3 Speech Therapy for Deaf Children

Prelingual deafness is a condition in which an individual experiences hearing loss prior to language acquisition.
It can be caused by a variety of factors, including *prenatal causes* (such as hereditary causes, infection or poisoning during pregnancy), *perinatal causes* (such as trauma during delivery) and *postnatal causes* (such as disease and infection) [13].
Prelingual deafness has different *degrees of severity*, ranging from mild to profound, and can be bilateral (involving both ears) or unilateral (involving only one ear). Audiometric tests can determine the degree of deafness and help choose the most appropriate treatment.
The choice of communication modalities and interventions will depend on the severity of the deafness, individual preferences and available resources.
People with prelingual deafness may benefit from early interventions, such as the use of hearing aids, cochlear implants, the use of sign language to facilitate communication, and speech therapy.

Childhood deafness affects children's communication, learning, and social and emotional growth [14]. Lack of hearing leads to difficulties in verbal communication and family relationships, limiting the perception of social roles. Deaf children interact mainly through the visual and tactile senses, with delays in the organizational function of language versus thought. The effects on language competence depend on the severity and timing of the onset of deafness. The lack of verbal input can cause delays and atypicality in the acquisition of spoken language.

Speech therapy helps these children develop age-appropriate language skills and improve their ability to communicate effectively with age-mates. For some deaf children, speech therapy focuses on developing oral communication skills, including speech articulation, pronunciation, and fluency. Therapists help children learn to accurately form and articulate sounds, words, and sentences. They also work on lip-reading and interpretation of facial expressions, improving the child's ability to

engage in oral communication.

Hence, we can see why it is essential to develop a structured approach to support these children. Firstly, the intervention must start early, preferably immediately after an early diagnosis of deafness, to maximize the opportunities for language and communication development.

A crucial aspect of defining the approach to be used is the **individualization of education**. Each deaf child is a unique individual with specific needs, different learning times, and personal preferences. Therefore, **education** must be highly customized, involving sign language educators, speech therapists, and other professionals, as well as parents, to adapt to the specific needs of each child [15].
Family involvement is a central pillar. Parents must be actively involved in the educational process, receiving training and support to help the child develop language and communication skills within the family environment. Social and psychological support is equally important, as deaf children may face unique emotional and social challenges.

Access to hearing technologies, such as hearing aids or cochlear implants, can significantly improve a child's hearing and should be an integral part of the process. Furthermore, the deaf children should be integrated into mainstream schools, with the support of specialized teachers and the involvement of speech therapists, sign language educators, and other professionals.

2.4 Education with Technology and Game: winning combination

The **education** of deaf children is a multifaceted field that necessitates a comprehensive approach to promote their linguistic, cognitive, social, and emotional development [16]. This requires collaboration between professionals and families to

create an environment conducive to their optimal growth. Family involvement is a key component, with parents receiving training and support to help their child develop language and communication skills. School teachers also play a crucial role in the upbringing of deaf children, particularly in speech and language development. Their collaboration with speech therapists and parents ensures that deaf children reach their full communicative potential. Teachers must consistently apply strategies introduced during speech therapy, both at home and in the school environment. Their constant presence in the child's life helps identify areas that need further attention or support, which can be communicated to speech therapists and parents for collaborative problem-solving.
To do this, parents, educators and professionals very often make use of resources that can be used to obtain a more functional approach. The two *resources* that are touched by Soundrise are *technology* and the *game*.

A resource with huge potential, which is gaining more and more ground in education, is the use of **technology** [17].
The integration of technology into education has become a significant topic in recent years, with the integration of technology offering both challenges and opportunities. Technology should not be seen as a mere tool but as a transformative medium [18]. A competent and proper use of digital technologies in education, with an appropriate methodological, didactic and organisational renovation, can really be able to support and help children in their study and growth, integrating it into the person's educational and training project.
The integration of technology into education has led to significant advantages, prompting educators to reconsider their pedagogical approaches. Educational institutions should invest time and resources in training their staff to use educational technology in a pedagogically sound manner. This commitment must extend to continuous training for both staff and students to ensure technology is used in a pedagogically viable manner.
The effective use of digital learning tools in classrooms can increase student engagement, help teachers improve their lesson plans, and facilitate personalised learning. Just think of how useful a computer or tablet can be for a child with specific learn-

ing disorders, when software is able to transform a long text to read into a speech synthesis to listen to, rather than a teacher's frontal speech into a concept map or infographic, or an evocative image to share with students who can also use it at home to study.

Assistive technology is any item, equipment, software program, or product system used to increase, maintain, or improve the functional abilities of individuals with disabilities [19]. It does not eliminate a student's learning difficulties, but it gives them the opportunity to work to their full potential and at the same level as their non-disabled peers. It can also increase students' sense of independence, as they are no longer overly dependent on teachers and parents.

Historically, Deaf students had limited access to education, primarily due to communication barriers. Technological advancements have provided Deaf students with remote access to education through virtual learning environments, online courses, and video conferencing [20]. These technological solutions have significantly improved the quality and accessibility of education for Deaf students.

These technological solutions have broken down communication barriers and provided equal opportunities for Deaf students.

The other resource used by Soundrise is **the game**. It is a solution that allows the world of education and technology to intertwine and perform better.

Game is the most congenial and spontaneous activity for the child: it is not "one activity" among others, but the main, if not exclusive, "activity" of childhood [21]. Modern and contemporary psychologists, neuropsychiatrists and educationalists agree in attributing enormous importance to the game as a diagnostic, emotional, affective and social factor. Every game reinforces and refines some physical or intellectual faculty. Through fun and perseverance, it makes easy what at first seemed difficult and stressful.

Games therefore have the same purpose as education: to help one easily understand something that was initially unknown. The use of games in education makes learning effective, motivating and engaging, facilitating the development of skills and knowledge in a lasting and memorable way. The playful side of learning reduces the stress associated with education and creates a less threatening learning environment.

Computer games, seen as learning platforms, can stimulate a child to increase his or her motivation in learning through a gaming experience.
Game-based learning (GBL)[22] refers to learning achieved through the use of games or video games, which may sometimes begin as entertainment tools but which are then used, with or without modification, to achieve an educational goal [23].
The game for deaf children is a very important factor that can help them to develop communication and interaction skills with other children. In these cases, parents often naturally tend to support the child in everything he or she does. Instead, games provide a safe context in which the child can take risks and learn adaptive strategies that are valid for adulthood, improving autonomy and self-confidence.
Game-based learning can help these children to explore their abilitie and to learn at their own pace. These computer-based speech training systems have the ability to offer children immediate and meaningful visual feedback [24]. An educational game created for deaf children should incorporate elements that take into account the specific needs of this population and promote inclusive and effective learning.

A very important feature of games is the feedback, usually immediate, that they always provide. Games produce clear feedback, allowing children to correct mistakes and improve their skills, and can be customised, ensuring customised learning [25].
Visual feedback is a crucial element in an educational game for deaf children. As these children mainly depend on visual communication, visual feedback is essential to ensure that they can understand and learn efficiently. It therefore uses signs and gestures, icons, animations, visual indicators and LIS text to communicate clear information, guide children and provide immediate responses. This customisable visual feedback helps deaf and dumb children connect visual signs with written language, making learning more effective and engaging.

2.5 Applications in this context

Throught the years, some applications were developed to support deaf child and deaf person. The most relevant and related to Soundrise will now be listed.

- visiBabble for reinforcement of early vocalization [26]

  VisiBabble is a real-time system that analyzes and processes a child's vocalizations, enhancing their language and cognitive development. It responds to syllables with colored animations and records acoustic analysis of speech. The system aims to improve children's vocal skills and increase the complexity and variety of sounds by enhancing syllable production over non-syllabic production. The prototype includes a laptop, microphone, and software that performs a point search function within a sentence, providing feedback, real-time visual responses, recording sessions, and collecting data on vocalization type and duration.

- SPECO, a multimedia multilingual teaching and training system [27]

  This research developed an audio-visual pronunciation teaching and training method and software system for hearing and speech-handicapped individuals. The project aims to create an audio-visual articulation training system for all languages, including English, Swedish, Slovenian, and Hungarian. The system uses visualized pictures of acoustic speech signals to help patients control their speech organs and generate most or all of the acoustic effects in speech. Collaboration from scientists from various fields, including digital speech processing, speech acoustics, linguistics, speech therapy, and technological facilities, is required. By comparing normal speech signals with disordered ones, patients can develop the ability to generate most or all of the acoustic effects in speech.

- Articulation Teacher App [28]

  Articulation Teacher is a mobile app designed by a speech therapist to enhance a child's articulation and pronunciation through games and at-home activities.

The app focuses on parent-child communication and incorporates target words into play and real-life context, rather than solely on drill practice. It features an intro video for each sound and a digital and printable cue card to help children pronounce sounds. The app is also designed to be used by speech therapists during sessions to aid in articulation practice.

- Little Bee Speech Apps [29]

  Little Bee Speech is the name of an app development company that believes that the most effective therapy comes from personal interaction with a therapist and/or parent. They strongly encourage the use of mobile devices and apps together with a teacher, therapist, or parent to help the child get the most out of the app and keep them on the road to success in reaching their therapy goals. Led by Speech-Language Pathologist Heidi Hanks, who has experience in early intervention and is the author of *Mommy Speech Therapy* [30], the company provides parents with tips and techniques to help their children improve their speech and language development.

- StorySign App [31]

  Huawei developed StorySign in 2018, the first AR app to translate a book into Sign Language in real time. This app aims to help deaf children overcome the challenge of reading, allowing them to explore the world of books. Children scan physical story books, revealing animations of character Star signing the relevant passage with the corresponding word highlighted. The app uses image and character recognition to identify words and translate them into the chosen sign language, providing a unique reading experience.

- Look to Speak App [32]

  Sarah Ezekiel, a patient with motor neuron disease, and her speech therapist Richard, in 2020, met with a small team from Google to explore how machine learning on smaller devices could make gaze-based communication technology more accessible to more people. Together they designed *Look to Speak* app,

which allows users to communicate, choosing from a list of phrases, using only eye movement and the phone's front-facing camera. It is designed to help people with motor disabilities and speech difficulties communicate more easily.

3 SoundRise: the background idea

This chapter explains the fundamental concept and overall functionality of the SoundRise application, detailing its past versions. Specifically, a focus will be placed on its reactivation based on models studied and defined inside the CSC (Centro di Sonologia Computazionale), and the procedural aspects of applying reactivation to SoundRise will be delineated.

3.1 What is SoundRise

SoundRise is an **educational game** that aims to support the child in discovering his or her voice. It is designed to complement a speech therapy treatment and to offer an autonomous learning instrument.
The aim of the SoundRise application, since its inception about 10 years ago, has been to create an interactive application that uses **real-time analysis of vocal features** to help children understand how their voice works.
This is particularly focused on deaf children, allowing them to explore and acquire a deep understanding of their vocal capabilities and monitor their progress through the graphical feedback generated.

The child's voice is interpreted by a sleeping **sun**. When the sun hears a sound (i.e. when the microphone recognises a sound), it opens its eyes and changes its appearance and position. The change of the sun depends on the changing sound

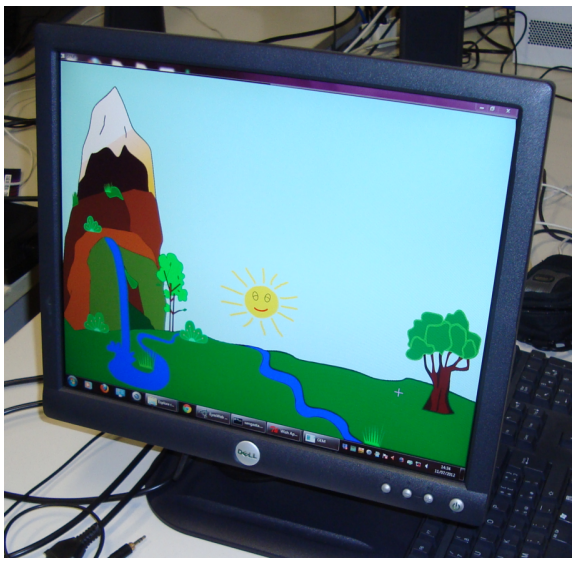


Figure 3.1: First interface of the SoundRise application developed in `Pure Data` on a PC in the CSC

expressed by the child's voice. The sun can rise and fall vertically according to the *pitch* of the sound emitted. It can grow bigger or smaller depending on the *sound intensity*. The sun also changes its colour, based on the vocal *timbre* detected. The recognition of this property is related to the identification of the five vowels of the Italian language: graphically, each possible timbre (5, as the vowels) corresponds to a different colour of the sun.

Therefore, the application uses the user's vocal input to create an interactive visual experience, providing to the child a visual feedback on the emitted sound of his/her voice, helping him/her to correct himself/herself, with the aim of educating him/her in the use of his/her own voice.

In Figure 3.1 it is possible to see the first interface of the SoundRise application, developed by Stefano Giusto [33] in `Pure Data` (more details in the next sections) that is running on a PC in the CSC.

3.2 The evolution of SoundRise over the time

The first idea for SoundRise was sketched out by Prof. Federico Avanzini, Prof. Antonio Rodà and Prof. Sergio Canazza from the Department of Information Engineering at the University of Padua, in collaboration with Dr. Serena Zanolla from the University of Udine. From then on, the project started to develop in the CSC (Centro di Sonologia Computazionale) at the University of Padua.

In 2012, the application was developed through two master thesis projects at the University of Padua, carried out by Stefano Giusto (Giusto S. Rodà A., *SoundRise: Studio e Progettazione Di un'Applicazione Multimodale Interattiva Per La Didattica Basata sull'Analisi Di Feature Vocali*) [33] and Marco Randon (Randon M. Avanzini F., *SoundRise: Sviluppo E Validazione Di un'Applicazione Multimodale Interattiva Per La Didattica Basata Sull'analisi Di Feature Vocali*) [34].
In these two master thesis was delineate a initial version of SoundRise as a standalone application for educational-therapeutic purposes aimed at a young audience as a support tool for vocal therapy or to practice voice modulation in singing.
In the master thesis of Stefano Giusto [33], the application was developed using the real-time graphical programming environment `Pure Data`[1] [35] and the external library *timbreID* for vocal timbre analysis. timbreID [36] represents a collection of externals for`Pd`, developed by William Brent. The features generated through the external objects can be used directly as control information in real-time performance. This library consists of a group of objects for extracting timbre features and a classification object that manages the resulting database of information. The incoming audio signal is processed by a specific object that detects attacks, defined as abrupt changes in the spectral envelope of the incoming sound, and generates a maximum match relation.
Subsequently, in the master thesis of Marco Randon [34], the application was modified to enhance its portability across different platforms. To achieve this, it became necessary to modify its architecture in a manner that would allow it to run inde-

[1] Pure Data (`Pd`) is an open source visual programming language for multimedia, developed by Miller Puckette in 1990s.

Figure 3.2: A screenshot of the SoundRise interface developed by Gabriele Turetta's thesis [37] as a web application with the Three.js [38] Javascript library and open-source software for 3D graphics and animations called Blender [39]

pendently of `Pd` while maintaining its core features and functionalities. Moreover, efforts were made to improve its graphical interface to adapt it for touchscreen devices. This new version of the application was developed in `C++` language and using the *libpd* library for the communication with`Pd`. In essence, libpd consists of variations of processing callback functions for different sample types, a set of functions for sending messages to`Pd`, and a series of function pointers for receiving messages from `Pd`. In order to receive messages, the client code must implement the appropriate receive functions and assign them to the corresponding function pointers in libpd. libpd includes language wrappers (`C++`, and others) that make libpd's functionality available to the respective programming language. These wrappers perform data type conversions between`Pd`'s custom data types and the standard data types of the target language. They provide an object-oriented interface for message exchange functions and function pointers. To receive messages from `Pd`, the client code must implement the necessary methods of the `PdReceiver` interface and register a receiver object using the methods of the `PdBase` class.

In the thesis of *Gabriele Turetta* (Turetta G. Canazza Targon S. Fiordelmondo A., *SoundRise 2.0: Sviluppo di un'interfaccia grafica interattiva in Three.js per supportare persone con disabilità uditive*) [37] a big improvement was done: a much more attractive and complex interface was developed using `Three.js` [38] and `Blender` [39], as it is shown in Figure 3.2. `Three.js` is a significant JavaScript library for managing 3D elements on the web. It was created to simplify and synthesize rendering of graphic elements in a web page. `Three.js` is built on `WebGL`[2], a web graphics library that offers lower astrasion levels compared to `Three.js`. Each project requires fundamental elements such as rendering, scene, geometry, material, texture, mesh, light, and camera. *Renderer* is the most important element, while *scene* defines the scene's layout and can be modified based on various parameters. *Geometry* is the combination of vectors that creates a scene's geometry, while *material* represents surface properties. *Texture* is the surface's appearance, while *mesh* is the mesh's inset. *Light* is the light that makes the meshes visible. *Camera* is an autonomous element that represents the scene in prosecutive or ortographic ways. Instead, `Blender` is an open-source 3D graphics software used for creating animations, models, and videos. Its versatile applications include architecture, design, animation, and cinema. `Blender`'s strength lies in its free and open-source nature, with a large community of users creating numerous plugins and scripts over the years. This evolution has made `Blender` a perfect competitor to industrial standards, which are not free and do not provide free educational support on the web.

A more detailed study of the functioning of SoundRise was carried out by *Riccardo Fila* (Canazza Targon S., Fiordelmondo A., Fila R., *SoundRise 2.0: Sviluppo di un modello di riconoscimento timbrico per un sistema di assistenza web dedicato a persone con disabilità uditive*) [40]. In this thesis, that is the last project related with SoundRise before this thesis, an in-depth study was carried out in order to develop a timbre recognition model. This project was developed in `Javascript` with the aid of `Web Audio API` (explained in detail in the chapter 4, on the section 4.2). Basically, a function calculates the first two formant frequencies of the voice and compares

[2] `WebGL` is the acronym of `Web-based Graphics Library`

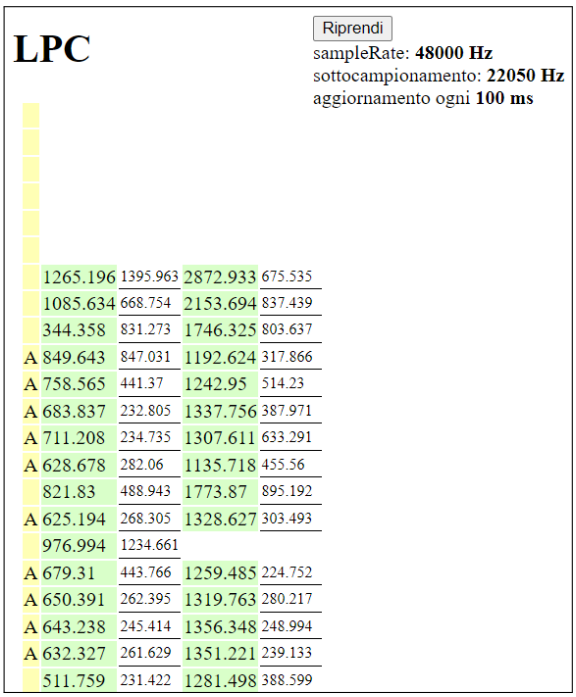


Figure 3.3: Interface of the web application created using Web Audio API of the vowel recognition model developed by Riccardo Fila [40]

them with those characteristic of Italian vowels in order to find an effective match and display it graphically. The idea of developing this type of application on the web was a turning point, as will be explained in Chapters 4 and 5, mainly because of the possibility of being able to exploit the potential of the `Web Audio API`.
In Figure 3.3, it is possible to see the interface of this prototype, so how it works: after starting the microphone, it analyses voice data and shows the resulting vowel (when it is able to capture it): the yellow column shows the vowel that was recognised by the model (if it was recognised), while the subsequent green columns show the analysed frequency values.

3.3 Reactivation

The preservation of interactive multimedia artworks is a complex and rapidly evolving field due to their complex nature and technological obsolescence. Current preservation strategies are inadequate as they do not address the complex relationships between analogue and digital components, their short life expectancies, and the experiences produced when the artworks are activated. The **Multilevel Dynamic Preservation model** [41], developed by the University of Padua's CSC, aims to preserve multimedia artworks through different levels of information, such as components, relationships, and activated experiences [42]. The model, that has been tested and optimized through case studies, allows for a process of change, minimizing loss of information about previous versions and shifting the focus from materiality to the experience and function that artworks activate. It records the transformation between reactivations, allowing for the inference of the artwork's authenticity.

The model focuses on the artwork as a complex, multi-media object defined by a process rather than fixed in time [43]. It uses horizontal and vertical layering to represent reproduction phases and exhibition relations, aiming to register the multimedia nature and dynamic authenticity of the artwork. This two-dimensional layering allows for easy zooming and understanding of the artwork's multifaceted nature.

3.3.1 In detail of the MDP model

The MDP introduces the **Digital Preservation Object (DPO)**, a digital file that encapsulates inter-related items in a logical architecture, representing all the items of a single exhibition of an artwork [44]. The items are all the analogue and digital elements that make up the artwork, the distinct elements of its score, and any other kind of documentation that testifies to the experience of each activation. The item level is the *smallest intellectually indivisible archival unit* [45]. The model

defines three different kinds of items with distinct functions for the artwork and with consequent different roles within the preservation model.
They can be classified as:

- *Components* are divided into hardware, software, audiovisuals, and various.
  - hardware including electronic devices and secondary items;
  - software including source code, apps, and utilities;
  - audiovisuals including video, film, sound, and photo materials;
  - the various category includes medatada schemes for components not listed in these categories, such as common objects, paintings, statues, and musical instruments.
- The *score* section provides metadata schemes for artwork scores, detailing algorithms and models used in DPO realization, operating instructions, and technical notes on individual components' functionality and linkage within the artwork.
- Metadata schemes for files including all of the DPO's *documentation*, including field recordings of audio and video, interviews, and images, are included in the documentation. These recordings pertain to the specific event or reactivation.

It is important to note that *score* and *components* items might be included in more than one DPO, and this entails that the components of the previous or original display, in whole or in part, will also be considered as new DPO elements.

The model highlights the multiple belongingness of items, which is applicable only to components or score type items, as documentation-type items must belong to a unique DPO. This property allows artwork to be represented as a process or dynamic object, not just a group of delimited records.
The Figure 3.4 is a graphic representation of the model. The dashed and linked items

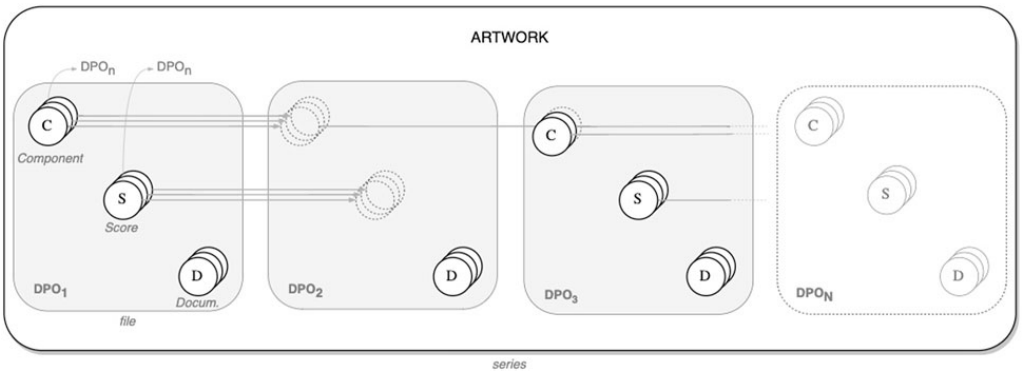


Figure 3.4: MDP model, developed by the CSC [41], aims to preserve multimedia artworks through different levels of information, using DPO, a digital file that encapsulates interrelated all the items of a single exhibition of an artwork in a logical architecture.

display the multiple belongingness property, where a specific component element or score can belong to several DPOs.
The model aims to promote a high degree of freedom in searching and managing information, allowing users to filter information and arrange narrative paths for various requirements.

The model was recently applied to a case study, the 1999 computer-based music installation *Il Caos delle sfere* by *Carlo De Pirro* [46], to test its efficiency in dealing with complex cases. The reactivation process produced new documentation and archived it in a database organized according to the MDP model. So, the MDP model has been shown to be efficient in dealing with complex cases, making it a starting point for defining a shared methodology for preserving complex artworks.

3.3.2 SoundRise reactivation process

In the CSC, the study of the preservation and reactivation of artefacts does not only relate to the artwork. The aim is to have a reactivation and preservation process that is valid for several areas, including the development (and maintenance) of applications. This is the case of SoundRise application: the intended goal was to perform **reactivation operations** to give this application the longevity and sustainability it deserves.

The reactivation process consists of five basic steps:

1. first of all, it is necessary to make a *collection* of the constituent elements of the project and the various stages of processing (in this case, from 2012 to the present), to gather as much knowledge as possible and get a clear overview of the dynamics.

2. the next step concerns the *assessment* of problems inherent in previous versions and/or prototypes;

3. as a third step, the *re-design* of the new reactivation is defined with attention to the longevity and sustainability of the application;

4. then the *implementation* phase of the newly defined redesign is carried out, obtaining a new working system;

5. the last phase is the *archive* phase, starting with the components developed in the previous phase, and producing documentation and instructions accordingly to it. The desired outcome will be a more sustainable DPO.

A complete resume of the reactivation process of the SoundRise application is shown in Figure 3.5.

Let's now go into more detail about these phases.

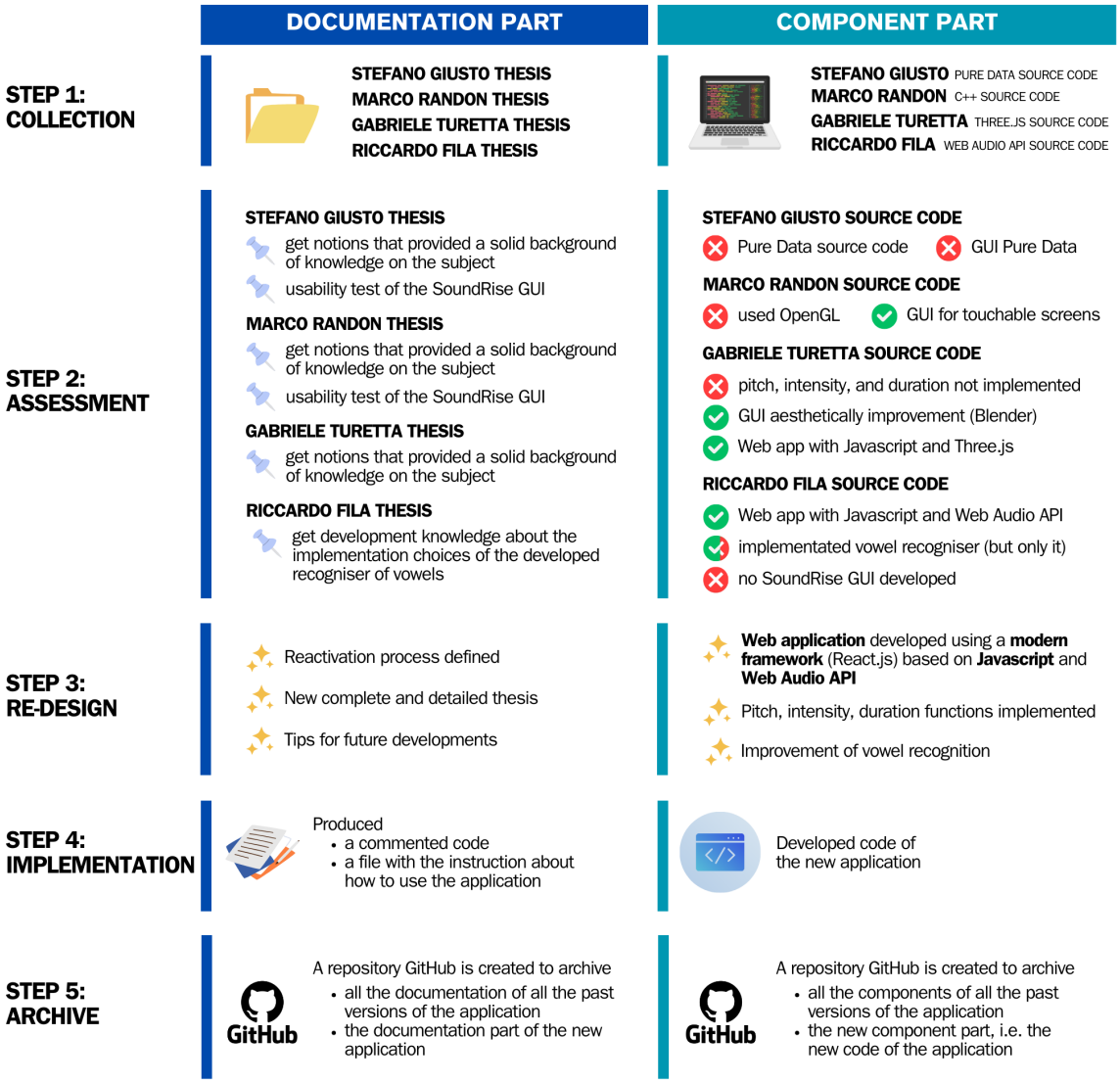


Figure 3.5: Schematisation of the five steps of the SoundRise reactivation process: for each step, a summary of the outcome for the documentation part and the component part is shown.

First step - the collection: the first step was to collect the elements to be analysed in order to proceed with the reactivation of the application.
Actually, not much documentation exists on this subject. The collected material is mainly *documentation* (details in 3.3.1 subsection), with Stefano Giusto's [33], Marco Randon's [34], Gabriele Turetta's [37] and Riccardo Fila's [40] theses, but all these thesis all these theses have also made a part of the source code available, i.e. the *component* part (details in 3.3.1 subsection). The foundation of the work was the two thesis projects of Stefano Giusto [33] and Marco Randon [34]. These two thesis represent the mainly part of the *documentation* (details in 3.3.1 subsection) of this project. Furtheremore, both of them, therefore, made a usability test of their application, with interesting insights. Regarding the *component* part (details in 3.3.1 subsection), the available source code is given by the application developed in `Pure Data` in Stefano Giusto [33] thesis, and the `C++` code project developed by Marco Randon [34].
In the thesis of Gabriele Turetta [37], the documentation part is less relevant than the component part: in fact, this thesis presents an source code based on `Three.js`.
The documentation part and the component part of Riccardo Fila's thesis [40], on the other hand, are somewhat more substantial. In fact, his thesis focuses in detail on the study and development of a new vowel recognition system.
The *score* part (details in 3.3.1 subsection), finally, comprehends all the high-level description of algorithms, such as the implemented audio signal processing (pitch extraction, intensity measurement, and vowel detection algorithms), and mappings between sound detection and the visual representation. In this case, the *score* overlap the documentation since each description has included in the previous theses.

Second step - the assessment: in the next step all these elements were examined, in such a way as to analyze the various problems and the various strengths that they presented, in order to start the process of reactivation and preservation of the SoundRise application.
Regarding the *component* part (details on 3.3.1 section), two non-maintainable systems were created by Stefano Giusto [33] and Marco Randon [34].

Stefano Giusto [33] He developed SoundRise as a `Pure Data` application, that gives a limited use of the application because of the limited number of devices that can support `Pure Data` and all the libraries that are necessary to install. The GUI was built using a visual patching language within `Pure Data` and it is not user-friendly. The code was replaced with different visual elements called *boxes*, which could be objects, messages, GUIs and comments, linked together. So, only the device that support `Pure Data` can run the application, and it restricts the use of the application. Instead, Marco Randon [34] developed an application in C++, which is no longer usable as it exploits the potential of a library which is no longer used. In fact, this interface, although it had the added feature of being suitable for touch-sensitive screens, was developed in `OpenGL` [47]. `OpenGL (Open Graphics Library)` is a cross-language, cross-platform application programming interface (API) for rendering 2D and 3D vector graphics. It is typically used to interact with a graphics processing unit (GPU), to achieve hardware-accelerated rendering. It is commonly used to make UI animations more responsive or to handle embedded video or to draw vector graphics. At present, however, `OpenGL` is increasingly being replaced by higher-performance systems and is no longer used: it cannot be said to be deprecated, as it is still supported by most platforms (except Apple's), it is simply no longer developed, and therefore is not updated in new technologies. This turns out to be a big limitation if the goal is to preserve an application over time.

In summary, in both theses, the interfaces that were developed appeared very simple, perhaps too simple, as they were not at all modern or appealing. Replicating it now would make the game too dated and lacking in enthusiasm.

A much better GUI was implemented in Gabriele Turetta's thesis [37], but he did not implement the various executive functions.

The most important source code is to be found in Riccardo Fila's thesis [40]. He only implemented the vowel recogniser functionality, but developed it with Javascript using Web Audio APIs (details in section 4.2), which allowed the application to be maintainable. In fact, his source code of the vowel recogniser was used in its entirety, with some slight improvements (details in section 5).

Regarding the *document* part (details on 3.3.1 section), all the thesis were read

carefully, in order to select all the technicall aspects that will could be useful in the future. Not so much the technical details used for the development were taken into consideration, but more all the notions that provided a solid background of knowledge on the subject. The Riccardo Fila's thesis [40] was most analysed for development knowledge, in order to understand the implementation choices of the recogniser system that he developed.
The other very interesting aspect in the documentation part, is about an usability test. In the Stefano Giusto [33] and Marco Randon [34] thesis, an usability test of GUI of the SoundRise appplication was conducted. The usability test helps identify potential issues in an application that may hinder task performance for average users or specific target groups. SoundRise, a tool for children to learn voice characteristics through graphical rendering, requires clear and unambiguous feedback. The test aims to measure the consistency of the system's response to children's voice input, ensuring that the children's voice can be used as an input for interaction with the application. The usability test was conducted at a Primary School in Gorizia to assess the consistency of SoundRise's response to children's voice input. The test involved a demo of the application, meaningful tasks for participants, a relatively homogeneous number of users (39 children), and an observer (Dr. *Serena Zanolla*). Every children first must produce a sound with their voice and observe what the sun does, and then answer two questions from the examiner:

- QUESTION 1: *What does the sun do?*
- QUESTION 2: *What do you think you do with your voice to make the sun rise/rise/change color?*

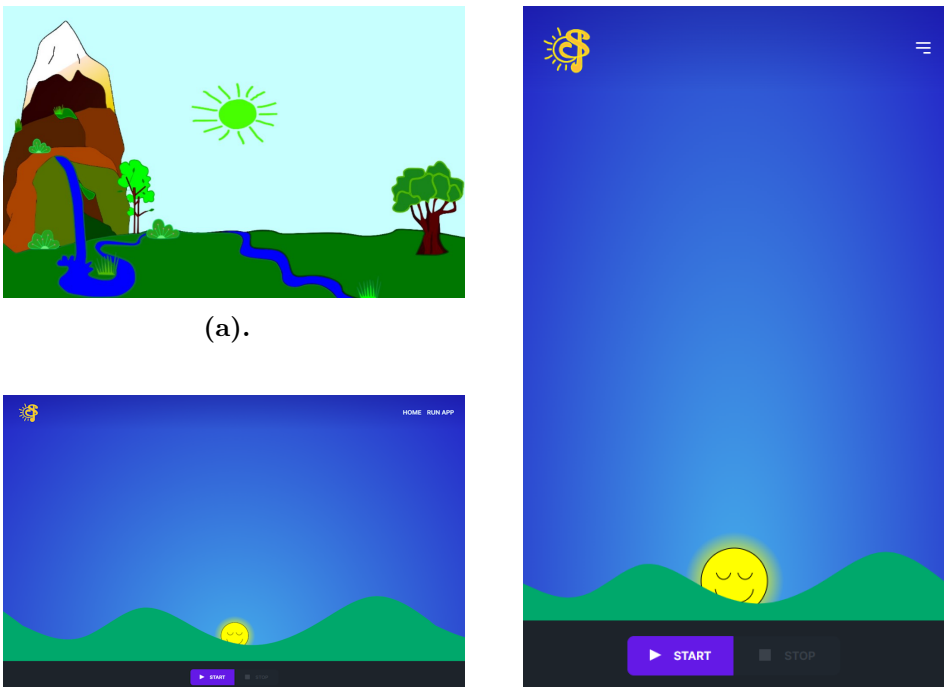
The data collected had no statistical validity due to the small number of participants. Results showed that children fully understood the movement of the sun (QUESTION 1) and the control of the graphic feedback through the features of one's own voice (QUESTION 2), except for timbre. In fact, less satisfactory results were obtained regarding the identification of the five vowels of the Italian language. This result was due to the lack of faithful identification of vowels, which does not lead to a

constant vowel-colour association, confusing the user. The lack of robustness of the database was considered the cause of the test results for the timbre feature.

Third step - the re-design: this important step deals with defining the design of the new reactivation with a focus on the longevity and sustainability of the application: the positive elements analysed in step 2 became basic ideas for enhancement, while the negative points were insights into where and how to improve.

For the *documentation* part (details on 3.3.1 subsection), this thesis represents a new complete and detailed version of the documentation. Inside of this thesis, all the details about the past version and the new version of SoundRise application are explained and analysed, the reactivation process that was applied was carefully outlined, and some ideas for future useful developments have also been added starting from the current version of SoundRise.

However, the largest part of the work concerns the *component* part (details on 3.3.1 subsection), both for the functionality of the application and for its graphic interface. Firstly, it was necessary to bring all the working code together into a single project and make the various parts communicate with each other. The code for phoneme recognition was stand-alone, as was the code for pitch and loudness detection. So first it was clear to put these different puzzle pieces together and make them fit together. The phoneme code developed by Riccardo Fila [40] has been best adapted for the purpose of the SoundRise application (details in chapter 5). Furthermore, the pitch and sound intensity detection functions have finally been developed (details in chapter 5) and merged with the SoundRise project.
The usability test made clear that the usability of the application is also linked to the way it is presented. If the application is accompanied by an explanation of the functionalities and how to interact, the user's understanding of each aspect of SoundRise increases and is more immediate, allowing the exercise and learning work to begin fully. Keeping in mind that one wants to create something that is as durable as possible, it was decided to focus on cutting-edge technologies that allow for fu-



(a). **(b).** **(c).**

Figure 3.6: Upgrade of the SoundRise user interface from the first to the latest developed version. In particular, (a) shows the original SoundRise user interface developed by Marco Randon's thesis [34], (b) shows the new SoundRise user interface, (c) shows how the new SoundRise user interface is responsive, simulating the screen size of a phone.

ture stability: hence the decision to abandon the development of a multi-platform application, and to embrace the idea of developing a **web application**. A web application is accessible from any device with a web browser, eliminating the need to develop different versions for various platforms and facilitating user access without the need for manual installation. In particular, web applications are highly scalable and can easily integrate with other web services, such as assistance management or interface frameworks.

In fact, with a web application, for the development of sound analysis functions, libraries from the web are used: they are more durable in any case, compared to software made to remain on the platform on which they were developed.

But above all, with the development of the web application, the goal of improving the interface was achieved. The reactivation of SoundRise as a web application allowed to recreate and modernize its interface using new and more user-friendly systems. One of the great new advantages is the fact that it is possible to apply techniques to

create interfaces that automatically adapt to the type of device on which they are viewed by the user. This feature is called responsive and is very important. Thanks to responsive design, the interface of a web application automatically adapts to the device used by the user, such as a computer, tablet or smartphone. For example, the Figure 3.6c shows how SoundRise is displayed on the screen of a smartphone. This screenshot could be obtained thanks to a Chrome extension called Responsive Viewer [48]. Responsive design enables an improved user experience and makes the website accessible in any device and any time. Regarding aesthetics, in the past versions a minimalist interface had been implemented, as shown in Figure 3.6a. Re-designing the interface by means of a web application makes it possible to always have a modern and attractive design that is easy to update and functional. Many famous applications (such as Facebook) have been developed with the same framework. Should this no longer be used in the future, a system will certainly be created to migrate from this system to one that is more modern.
Furthermore, as the goal is to create a *game-based learning*, actively engaging children and motivating them to spend more time on learning, it was decided to keep the minimalistic design in order to result a simple and engaging, as shown in Figure 3.6b. This design aims to transmit, attract, and retain user attention, facilitating their understanding of the represented information. Creating a safe and familiar environment allows users to view this application as a friend rather than a enemy.

All these solutions will allow the SoundRise application not to lag behind the times, but to always be up to date, that is the main goal of a reactivation process.

Fourth step - the implementation: as indicated by the name of this phase, the implementation phase corcerns all the operations of development of the application, following the guidelines achieve with the re-design phase.
For more in-depth information on this phase, refer to Chapter 4 and Chapter 5: in them, this phase has been carefully described and explained in detail. In particular, in Chapter 4 after providing a background theoretical knowledge on sound and the features relevant to the SoundRise application, a detailed description of the

technologies used in the development of SoundRise will be presented and analysed, explaining why they were chosen, while Chapter 5 outlines the functionality of the SoundRise application, illustrating how it was implemented, the various mechanisms that are generated based on user actions, and the interface modifications.

Fifth and final step - the archive: a *model of archivation* is the structured approach or set of practices followed for archiving an application. This could involve the systematic preservation of the application's codebase, dependencies, documentation, and associated data to ensure future reference, compliance, or potential reactivation. Developers use various strategies for archiving applications, including Version Control Systems (VCS), code repositories, documentation, containerization technologies, backup practices, deprecation planning, cloud services, legal and licensing considerations, updating dependencies, and staying informed about evolving best practices. These strategies help track changes in code, provide insights into architecture, design decisions, dependencies, and setup instructions, and ensure consistent deployment and deployment.

The goal of the creation of a performant model of archivation is to establish a reliable and organized framework for maintaining the integrity and accessibility of the application and its related assets over time.
This phase of the SoundRise application reactivation process is concerned with the definition and use of an archive model that can preserve and sustain the application over time. Introducing a robust model for archiving applications is essential for developers seeking efficient version control, collaboration, and project management. One such powerful model is exemplified by **GitHub**, a web-based platform and version control system. GitHub offers a comprehensive suite of tools that streamline the archiving process, providing developers with a structured and collaborative environment for preserving their application code. From version tracking to documentation, issue management, and community engagement, this archiving model on GitHub is designed to enhance the development lifecycle.
GitHub's archiving model is structured and efficient, while its version control system

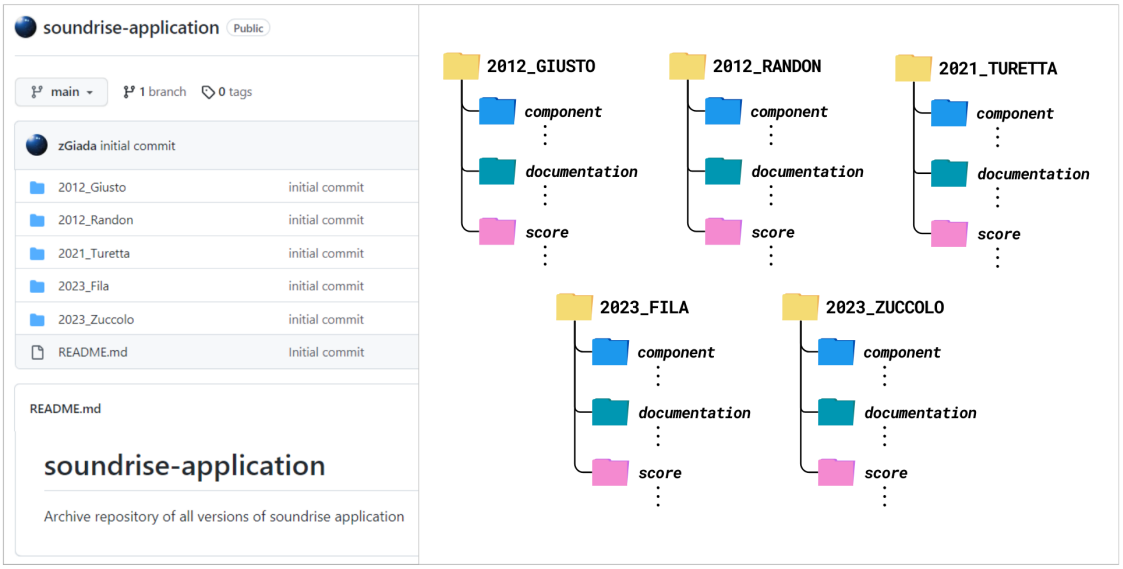


Figure 3.7: Model of archiviation of SoundRise application into GitHub, through the subdivision into folders with the version of the application and their relative three subfolders (component, documentation, and score).

allows for easy tracking of changes. It simplifies collaboration by allowing seamless contributions through forking and branching mechanisms, allowing for documentation of code, and managing bug reports and feature requests. It also integrates with CI/CD services, automating essential processes like building, testing, and deploying applications. As a remote repository, GitHub serves as a secure backup for code, allowing easy recovery in case of local machine failures or data loss.

For the reasons outlined above, the GitHub platform was used for the archiving of this project. The GitHub repository with the SoundRise version archive is available at the following link: `https://github.com/zGiada/soundrise-application`.

For each versions of the Soundrise application, folders have been created with the name of the version. The organization of the application into version-specific folders with designated sub-folders serves several key purposes for effective archiving and project management.
Each version is housed in its own folder, simplifying retrieval and reference. This structure aligns well with version control systems, allowing for independent commits and revisions.

Within each folder there are 3 sub-folders called *component*, *documentation*, and *score*, as it is shown in Figure 3.7. For each folder with the version of SoundRise, the sub-folder *component* will contain the code part of that version and elements related to it, the sub-folder *documentation* will contain the documentation part of that version, and the sub-folder *score* will contain the score elements that belongs of that version.

This hierarchical structure eases navigation through different versions and simplifies the process of finding relevant components, documentation, and score elements. It also streamlines archive and backup management, making it straightforward to create backups or transfer the project. The organized folder structure supports collaboration by enabling team members to work on specific versions without conflicts. Additionally, it aids in sharing specific versions of the project with collaborators or stakeholders. Over time, the structured archive promotes maintainability and longevity, reducing the risk of errors, enhancing traceability, and simplifying troubleshooting if issues arise during the evolution of the project.

4 Used Technologies

In this chapter, after having provided a theoretical basis of sound in the characteristics that affect the SoundRise application, all the technologies used for the development of the SoundRise application will be described and discussed, motivating their choice.

4.1 Theory of Sound

Before starting to go into details about technologies, this section will give a general overview of the world of sound, focusing on what is most pertinent to the development of the SoundRise application.

In physics, **sound** is a form of energy known as acoustic or sound energy. Sound is created by the vibration of an object, which causes a movement of particles, which propagates through the air in the form of a longitudinal wave. The waves of particles cause the eardrum of the ear to move back and forth, this movement is then transformed into nerve impulses (electro-chemical), processed by the brain, which finally gives rise to the auditory experience. When a vibration passes through one complete up-and-down motion, it has completed one cycle. The number of cycles that a vibration completes in one second is expressed as its **frequency**. If a vibration completes 50 cycles per second, its frequency is 50 Hertz (Hz[1]); if it

[1]Hz stands for Hertz, the unit of measurement of frequency, equals 1 cycle per second.

completes 10,000 cps, its frequency is 10,000 Hz, or 10 kiloHertz (kHz). Every vibration has a frequency, and humans with excellent hearing may be capable of hearing frequencies from 20 Hz to 20,000 Hz. However, the limits of low and high frequency hearing for most humans are about 35 Hz to 16,000 Hz. Frequencies just below the low end of this range, called infrasonic, and those just above the high end of this range, called ultrasonic, are sensed more than heard.
Psychologically, and in musical terms, humans perceive frequency as **pitch** ness of a sound. The pitch is the characteristic of the sound that depends on the frequency of the sound wave that generated it, and it permits to distinguish whether a sound is high or low. The more times per second a sound source vibrates (therefore, as higher the frequency), the higher its pitch. Consequently, the lower the frequency value, the lower the pitch value. Middle C (C4) on a piano vibrates 261.63 times per second, so its fundamental frequency is 261.63 Hz. The A note above middle C has a frequency of 440 Hz, so the pitch is higher. The fundamental frequency is also called the first harmonic or primary frequency. It is the lowest, or basic, pitch of a musical instrument. The range of audible frequencies, or the sound frequency spectrum, is divided into sections, each with a unique and vital quality. The usual divisions are called octaves: an octave is the interval between any two frequencies that have a tonal ratio of 2:1. The range of human hearing covers about 10 octaves. Starting with 20 Hz, the first octave is 20 Hz to 40 Hz; the second, 40 Hz to 80 Hz; the third, 80 Hz to 160 Hz; and so on.

However, regarding the frequencies of the possible sounds emitted by humans, the discussion is different. At its most basic level, the human voice acts like a wind instrument. The lungs expel air through the vocal cords, which vibrate and act like a double reed. The voice is then radiated through the larynx and the mouth and out into the air. Sound is modified by the movement of the tongue, jaw, and lips and the rate of exhalation. Pitch is determined primarily by the vocal cords and the larynx, but mouth, lip, and tongue movements change timbre. The wide-ranging and complex harmonics and overtones of the human voice are a product of the mouth, nasal cavity, and chest cavity.
The frequency range of the human speaking voice, compared with that of some

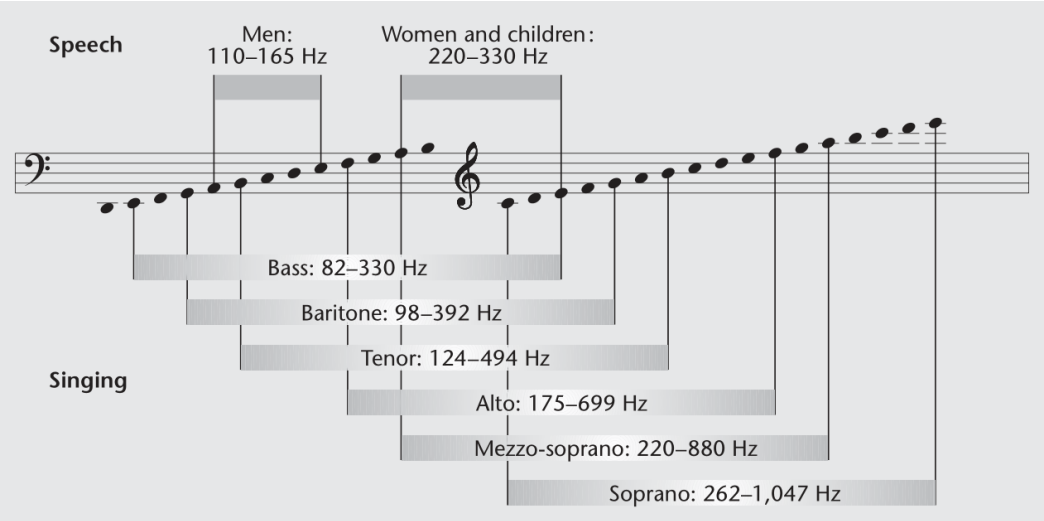


Figure 4.1: Fundamental frequency ranges for speech and singing in the human voice for men and, for women and children

musical instruments, is not wide. Fundamental voicing frequencies produced by the adult male are from roughly 80 Hz to 240 Hz; for the adult female, they are from roughly 140 Hz to 500 Hz. But one thing must be pointed out: the human voice is a complex instrument with narrow ranges, but its harmonic richness makes it a highly complex instrument. Frequency content varies with vocal effort, with loud speech increasing midrange frequencies and reducing bass frequencies, while quiet speech reduces midrange frequencies and lowers overall pitch. So, harmonics and overtones carry these ranges somewhat higher. This applies even more to the singing: although the speaking voice has a comparatively limited frequency range, the singing voice does not. The singing voice is capable of subtle and almost unlimited variations in pitch, timbre, and dynamic range. In summary, ranges for the singing voice are significantly wider as it is possible to see in Figure 4.1.

Vibrations within objects induce the movement of molecules, generating pressure waves at specific rates of alternation, thereby establishing the frequency. These vibrations dictate the extent of displacement in molecules, propelling them from equilibrium to the peak (crest) and trough of a wave. The intensity of a vibration directly influences the number of displaced molecules, with more intense vibrations leading to a greater displacement. Consequently, the number of molecules in mo-

tion, and therefore the size of a sound wave, is called **amplitude**. The rate at which sound energy reaches a given cross-sectional area is known as the **sound intensity**. Human hearing encompasses a broad range of intensities, prompting the use of a logarithmic scale called the **Decibel Scale (dB)** to express sound intensity. The decibel, being a dimensionless unit, lacks a specific physical quantity and is employed as a comparative tool for ratios between quantities like acoustic energy, sound pressure, and electric energy such as power and voltage.
The subjective perception of sound intensity, or amplitude, is known as **loudness** or softness. Loudness is an individual's subjective assessment of a sound's volume, whereas sound intensity can be objectively measured using the Decibel Scale. The relationship between the intensity of a sound and its decibel rating follows a mathematical pattern, increasing by 10 dB for every tenfold rise in intensity. Thus, the decibel rating serves as a quantifiable indicator of a sound's perceived loudness or softness, providing a standardized means of comparison.

Sound is often depicted as a single, wavy line, known as a sine wave. It is a pure tone, i.e. a single frequency devoid of harmonics and overtones. Most sound, though, consists of several different frequencies that produce a complex waveform, which can be seen, for example, on test equipment and digital editing systems and in spectrographs. Each sound has a unique tonal mix of fundamental and harmonic frequencies that distinguishes it from all other sound, even if the sounds have the same pitch, loudness, and duration. This difference between sounds is what defnes their **timbre**, i.e. their tonal quality, or tonal color. Timbre is notoriously difficult to define, describe, and measure: all things being equal, i.e. perceived pitch, loudness, etc., timbre is the quality that makes a clarinet sound like a clarinet, a violin sound like a violin, and the two sound different from each other, and the same applies to human voices. Timbre can be thought of as the auditory equivalent of color. Timbre itself is not measurable, i.e. it is not an intrinsic property of a sound, but rather, a perceptual one. However, it is related to certain physical characteristics of sound that are measurable, such as the spectrum [49]. **Formants** are concentrations of energy in particular areas of the spectrum and are usually seen as broad peaks (local maxima). They are particularly useful in the analysis of vocal timbre: the

general distinction between different vowels and the characteristic timbre of different people's voices, and those of men, women, and children, can be described in terms of their formant structure. In the voice, formants are the result of resonant structures in the vocal tract. A particular set of formant frequencies characterize each vowel and are relatively independent of a voices pitch. Female and male voices obviously have different formant frequency ranges, however the ratio between formant frequencies is consistent across males, females, adults, and children.

All these features (and many more) are extensively described in the book *Audio in Media* by *Stanley R. Alten* [50]. In the next chapter, it will be explained how these features (pitch, sound intensity and timbre) are captured, analyzed and displayed graphically to the user of the SoundRise application.

4.2 Web App with Javascript and Web Audio API

The Soundrise application is developed as a web application. A web app is software accessible from any browser, created using languages such as JavaScipt, HTML or CSS. Unlike mobile apps, designed for a specific platform, such as iOS or Android, web apps allow you to access directly from the browser and can be used on any device as it adapts better to the device on which it is used, preserving a very similar to a mobile application. The language predominantly used for this project is **Javascript**. JavaScript is a high-level programming language used for client-side execution of scripts embedded in HTML pages or javascript extension files. It allows manipulation of style, content, and user interaction. Its high level abstraction makes it easily comprehensible and has enabled the development of dynamic and interactive applications on the Web. In fact, it permits to create the logic of the user interface and to exploit the APIs (Application Programming Interface) provided by the browser: from mouse management to image manipulation, from dynamic data requests to local data management. An API is a portion of code that a programmer is given access to, which controls a larger unseen body of code within certain con-

straints. Imagine if, in order to learn how to play your favorite musical instrument, you had to literally build it from scratch. As you can imagine, this would be very troublesome. Thus, it's much more convenient to learn to play a premade musical instrument. In a similar way, programmers write APIs that expose only small pieces of code for developers to use, and these small pieces of code allow you to do a lot of work with minimal effort. Thanks to the numerous APIs made available by browsers, JavaScript is an extremely high-performance language.

Soundrise exploits the **Web Audio API** [51], a high-level JavaScript API that allows sound to be played, manipulated and synthesised within a Web application. Supported by all major browsers except Internet Explorer 11 and Opera Mini, it is a very powerful tool that works by generating sounds from scratch or by manipulating existing audio files or a sound source via line input.
Web Audio API presents a new interface model that compensates for the deficiencies of HTML5's `<audio>` element [52]. In fact, for the management and manipulation of audio on HTML documents, the `<audio>` element was created by HTML5, which is recognised and supported by all modern browsers: it would be very useful as it does not require plug-ins, but it has many limitations, especially with regard to the implementation of interactive applications, which have not allowed it to be used for this purpose [53]. The strength of the Web Audio API was to introduce interactive audio to the Web, which is also suitable for applications with non-predictable behaviour, such as video games, where sounds can interact with the environment and be mixed in a complex way, managing effects through filters and managing the positioning of sounds in space.

The Web Audio API provides for the handling of audio operations [51] using `audio context`. The `audio context` is a constructor that returns an object containing all of the methods and properties usable from Web Audio API. Technically, the `audio context` is an oriented graph of audio nodes (`AudioNode`) that defines how the audio stream flows from the source node (`SourceNode`) to the destination node (`DestinationNode`). Connected nodes perform the desired audio operations, forming a modular structure that offers the flexibility to create complex audio functions.

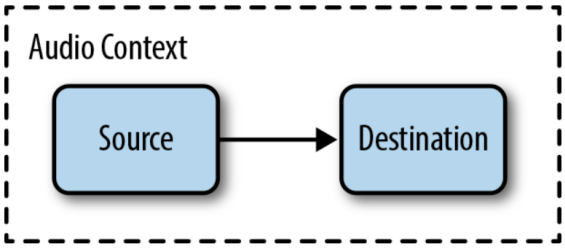


Figure 4.2: The `audio context` is a directed graph of audio nodes that defines how the audio stream flows from its source to its destination. As audio passes through each node, its properties can be modified or inspected [51].
The simplest audio context is a connection directly form a source to a destination node.

The outputs of these nodes can be connected to the inputs of others, which mix or modify these streams of sound samples. Once the sound has been sufficiently processed and the desired effect has been achieved, it can be connected to the input of an output (destination) node, which saves the result or sends the sound to speakers or headphones. The `source node` has no input as it is the sound source, while the `destination node` has no output as it is the last node in the graph that can be imagined, in an abstract way, as the computer speakers.
The simplest audio context is a connection directly form a source node to a destination node (Figure 4.2). It is therefore possible to create even very complex graphs in which several modules, in sequence or in parallel, work together for the management and manipulation of the entire audio aspect of an application. This is because the `audio context` is based on *modular routing*: each node can be arbitrarily connected to other nodes of different types and/or, more specifically, can be connected to `AudioParam` parameters that control the behaviour of other different `AudioNodes`. Each node is characterised by a set of `AudioParam` parameters that control specific aspects of their functionality: it can be set to a specific value or scheduled for precise times, allowing for arbitrary automation curves.

In summary, the workflow for audio processing that Web Audio adopts starts with the creation of the `audioContext` instance that gives access to all interface functionality. Next, one must create the audio sources within the `audioContext` (`SourceNode`) and add the nodes that modify and analyse the audio according to the desired parameters (called `EffectNode`). The last step is to choose the final destination of the

audio (i.e. the `DestinationNode`), e.g. the audio speakers of the user's terminal, and connect the source nodes to zero or more effect nodes and then to the chosen destination node. In the next chapter it will be explained in detail how web audio API is used in SoundRise application.

4.3 Development of the interface with React JS

Today, front-end frameworks and libraries are becoming an essential part of the modern web development stack.

React.js [54] is a front-end library that aims to simplify the intricate process of building interactive user interfaces. It has gradually become the framework of choice for modern web development within the JavaScript community. Since its debut in May 2013, the library has grown to become one of the most widely used front-end libraries for web development. Developed by Facebook software engineer Jordan Walke, React has revolutionised web development by simplifying the development process and introducing reusable components, transforming the approach to web development. In fact, in React, applications are developed by creating reusable components that can be viewed as independent Lego blocks. These components are individual pieces of a final interface that, when assembled, form the entire user interface of the application. The main role of React in an application is to manage the view layer of that application just like the *V* in an MVC (model-view-controller[2]) model by providing better and more efficient rendering. Instead of managing the entire user interface as a single unit, React.js encourages developers to separate these complex UIs into individual **reusable components** that form the building blocks of the entire UI. Imagine a user interface created with React as a collection of components, each responsible for the output of a small piece of reusable HTML code. In this way, the ReactJS framework combines the speed and efficiency of JavaScript with a more efficient method of DOM manipulation to render Web pages

[2] Model–view–controller (MVC) is a software design pattern, usually used for web applications and desktop GUIs, that divides program logic into three interconnected elements: Model, View, and Controller.

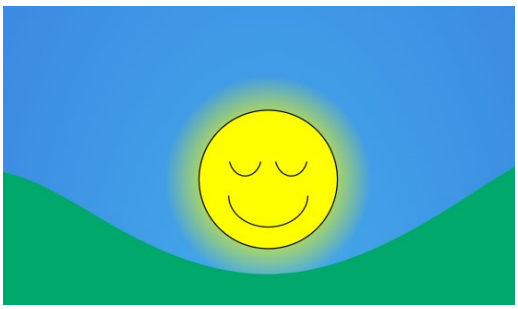


Figure 4.3: How the sun appears when the `SunSleep` component is selected.

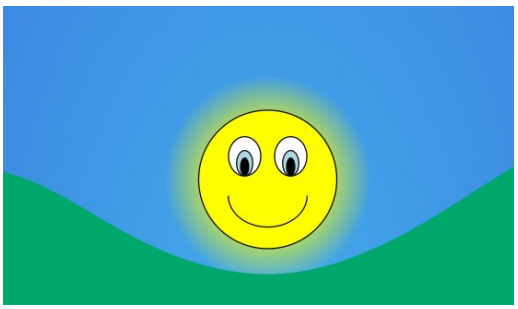


Figure 4.4: How the sun appears when the `SunAwake` component is selected.

faster and create highly dynamic and responsive Web applications.

What was developed for the SoundRise application, based on React.js, is a Single Page Application (SPA). It is a client-side application loaded entirely by the client and does not require a back-end server: the application is dynamic enough to retrieve and render screens that meet users' requirements as they navigate within the application itself. So, there is a main file that is updated in its content using JavaScript, interacting with user actions. This **main file** is called `play.jsx` and it is the core of the Soundrise application. It is the interface that the users use to play the game and where they receive the visual feedback. It is possible to say that `play.jsx` is the first component of this application.

The firsts two important components that are part of this application are those that determine the *state* of the sun element.
Before explaining them, it is important to emphasise the word *state*: in React.js, *state* generally refers to data or properties that need to be tracking in an application. React's state management within components is a key feature: the introduction of hooks has made it even easier to handle state and side effects in your functional components. The `useState` hook is a simple yet powerful method that returns an array with the current state and a function to update it. It takes an initial state value as an argument and returns an updated state value when the setter function is called. In this project, `useState` is implemented with the `useEffect` hook, which provides the handling of side effects such as retrieving data or updating the DOM.

In SoundRise application, the sun, the object used to represent voice feedback, can have two *states*: `sunSleep`, i.e. when the microphone hears no sound, and `sunAwake`, i.e. when the microphone recognises the sound of a voice. These two states represent the two components that have been created to handle sound recognition.
In the `sunSleep` component, a sun with a smile and closed eyes was drawn, as Figure 4.3 shows (via several overlapping SVG[3]). Ideally, when the microphone does not pick up any sound, the sun does not *rise*, so it remains asleep.
This state is represented by the sun's closed eyes.
In the `sunAwake` component, the sun drawn (again by several overlapping SVG's) has a smile and open eyes and a sunny expression, as Figure 4.4 shows.
When the microphone recognises a sound that is associated with the voice (i.e. not background noise), the sun changes from asleep to awake, i.e. *rises*.
This state is represented by the sun's open eyes and the fact that it begins to change its appearance and position.
The role of these two components is to show to the user (through a very simple graphical feedback) when the application recognises a vocal sound and when it does not. In order to change the sun's aspect according to the sound of the voice being heard, external libraries have been developed in Javascript and are called by `play.jsx` whenever a vocal sound is recognised. These libraries are discussed in detail in the next chapter.

The others components that are created for this applications are three: the `Header`, the `Footer` and the `Scenes`.
The `Header` component was created to have a unique header for the application page. Inside, it contains the SoundRise logo and the horizontal navigation menu. The navigation menu allows the user to move through the pages of the application. There are three pages that the application presents: Home, Run App and About.
The **Home** page provides an overview of the SoundRise application: explains what it is, how it works and who developed it.

[3]A SVG (Scalable Vector Graphics) file is vector-based formats used for displaying two-dimensional graphics, charts, and illustrations on websites, which can be scaled up or down without losing any of its resolution.

The **Run App** page contains the game: it is on this page that the user can play with the developed application.
The **About** page explains the possible feedback that the SoundRise game can give, so that the user can understand how it works.
This navigation menu has been programmed to be responsive: when the application is run inside a device with a reduced screen, the menu is closed into a hamburger menu. When the user clicks on the hamburger, the drop-down menu with the various page items opens.

The `Footer` component was created to have a unique area to close the page. Its content changes according to the page you are on, but the area occupied always remains the same.
On the Home and About pages, the Footer shows the copyright of the application, referring to the CSC and this master thesis study.
On the Run App page, the footer takes on a different task: it is in this area that the start and stop buttons are inserted to start or stop the microphone. Here, there is a *button-group*, with alternative buttons.
In the initial state, the active button is the stop button, since when the application is started, the microphone is switched off, and therefore the only possible action is to switch the microphone on via the start button. The start button is therefore the only button enabled, i.e. that can be clicked. Once the start button is clicked, the active button becomes the start button, as the microphone is now on and therefore the only possible action is to turn the microphone off via the stop button. At this moment therefore the stop button is the only button enabled, i.e. that can be clicked. And so on. So it can be understood that the *active button* depends on the *state* (`useState`) of the microphone, while the *clickable button* depends on the *state* (`useState`) of the button (clickable or not).

The last component to be created is called `Scenes`. Through this component, the background layer of the scene can be created. Currently, it is a green SVG that repeats horizontally across the width of the page, creating a background of a hilly landscape. The `Scenes` component was created not so much to reuse the background

format on several pages (although this is actually its current use), but to be able to change the landscape in the future. At present, a hilly landscape is preset, but in this way, the functionality can be developed in the future to allow the user to change the panorama by means of a setting. With this component, one would not have to change the entire code, but only that which concerns the component itself.

5 How SoundRise works

This chapter describes how the SoundRise application works: it explains all the various scenarios in which the SoundRise application can be found, showing which mechanisms are generated and which functions operate consequently and how the interface changes in each of them.

5.1 Overview of SoundRise Functionality

As already described in the previous chapters, the soundrise application has a minimalist but intuitive interface. In this version of SoundRise, it was decided to present a hilly landscape: in fact, the sun is located at the bottom of the screen, behind a hilly landscape, waiting to rise as soon as it hears the sound of a voice. For the representation of the sun, the goal was to achieve an ultra-simple layout (in 2D) that could communicate the joy and sunny disposition that the sun generally brings, especially to children: therefore, through the interlocking of SVGs (as explained in Chapter 4), the sun always appears with a smile, even when its eyes are closed and it has still to rise. The sky in the background is colored in gradations of blue, with a brighter light near the sun, and a darker color in more distant areas.
At the base of the hilly landscape, there is the footer bar with two buttons for switching the microphone on and off in the center. Basically, the operation of the SoundRise application depends on which of the two buttons is active (i.e. the *state* of the microphone), and on the resulting graphical result.

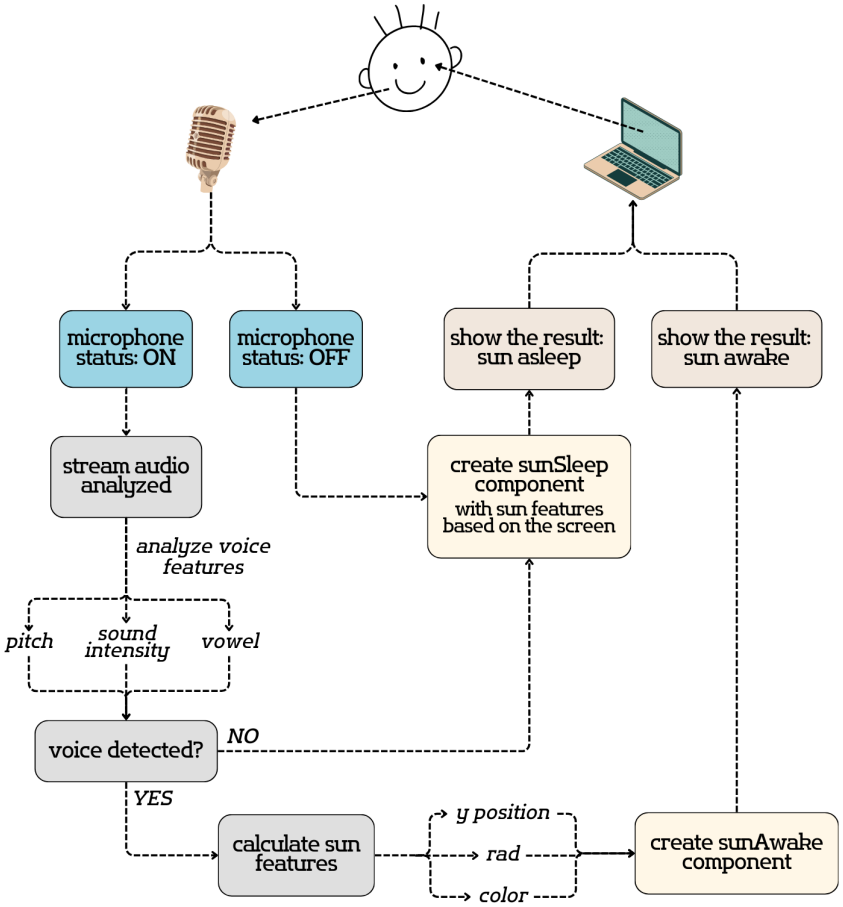


Figure 5.1: The possible workflows of the SoundRise application: from the two initial situations relating to the state of the microphone (on/off), to the two final situations relating to which of the two components (`sunSleep` or `sunAwake`) is displayed as a result, depending on whether the voice is detected or not.

In fact, as can be seen in Figure 5.1, two *initial situations* can be defined, which are:

- `microphone status: on`;
- `microphone status: off`.

The two *final situations* correspond to the graphical result obtained:

- `show sun asleep` - i.e. voice not detected
- `show sun awake` - i.e. voice detected with defined characteristics.

5.1.1 Handling of microphone and visual feedback

From a logical point of view is quite simple to understand that the Start Button turns on the microphone, while the Stop Button turns off the microphone.

From a practical point of view, everything starts with these two lines of code:

```
1 const [startButtonDisabled, setStartButtonDisabled] = useState(
    false);
2 const [stopButtonDisabled, setStopButtonDisabled] =  useState(true)
    ;
```

startButtonDisabled and stopButtonDisabled are state variables, and setStartButtonDisabled and setStopButtonDisabled are their corresponding state-setting functions, managed by the `useState` hook. They are used to control the disabled state of the Start and Stop buttons, respectively. The boolean value represents whether the Start button should be disabled or not (and the same for the Stop button).
Simply, they help manage the user interface to prevent unintended interactions during specific states of the application.
But their intervention is fundamental. In fact, they work together in order to represent (and to set) the state of the microphone.
When the boolean value of the Start button is set to `true`, the Stop button is always set to `false`: so the Start button is disabled and the user cannot press it, while the Stop button is active, so the user can only press that button. This means that at that moment the microphone is turned on, because the only action that the user can do is to press the Stop button, which permits to turn off the microphone.
And vice-versa, when the boolean value of the Start button is set to `false`, the Stop button is always set to `true`: so the Stop button is disabled and the user cannot press it, while the Start button is active, so the user can only press that button. This means that in that moment the microphone is turned off, because the only action that the user can do is to press the Start button, which permits to turn on the microphone.
Therefore, to make it immediately clear which buttons the user can press, the two

buttons are colored with two different colors: the clickable button is purple, while the active one (i.e. the disabled button) is black with the text color semi-transparent.

This management of the microphone and the Start and Stop buttons is done through two event handler functions: handleStartListening() and handleStopListening(), declared as follows:

```
const handleStartListening = () => {
    setIsListening(true);
    setStartButtonDisabled(true);
    setStopButtonDisabled(false);
};

const handleStopListening = () => {
    setIsListening(false);
    setStartButtonDisabled(false);
    setStopButtonDisabled(true);
};
```

First, let's take in exam handleStartListening() function.
With handleStartListening(), the goal is to initialize the audio listening functionality. In addition to setting properly the startButtonDisabled and stopButtonDisabled states, it set another variable called isListening, which was previously declared like this:

```
const [isListening, setIsListening] = useState(boolean_value);
```

isListening and setIsListening are used to control whether the audio is actively being listened to or not.
When the Start button is clicked, the `handleStartListening()` function sets isListening to `true` using setIsListening(true) and triggers the initialization of audio starting the listening process.
The function `handleStopListening()` is called when the Stop button is clicked because the goal is to stop the audio listening functionality: in addition, to setting properly the startButtonDisabled and stopButtonDisabled states, it sets the state vari-

able isListening to `false` using setIsListening(false), in order to mark the end of the audio listening process and performs cleanup actions (more details later).
Basically, this hook permits the initialization of audio-related processes, such as obtaining user media (turning on the microphone) and starting the audio analysis loop, in order to get the voice feature (more details in the next chapters).

The last hooks correlated with the use of the microphone are sunListen and setSunListen, which were previously declared like this:

```
1 const [sunListen, setSunListen] = useState(boolean_value);
```

They are used to control the visual representation of the sun in the interface, i.e. which component is called. That means that the value of sunListen (managed by setSunListen()) is set to `true` only when the sound of a voice is detected. In fact, the value of sunListen is used to determine which component should be rendered, when the sun must be awake, sunListen is set to `true` and the `SunAwake` component is called; on the other hand, when the sun must be asleep, sunListen is set to `false` and the `SunSleep` component is called, as it is shown in this piece of code:

```
1 { sunListen ? (
2   <SunAwake />
3 ) : (
4   <SunSleep />
5 )}
```

In conclusion, the Start and Stop buttons, through the handleStartListening() function, the handleStopListening() function, and the value of the variable isListening (set with setIsListening() function), allow the two possible *initial situations* (defined in 5.1) of the microphone to be managed; while sunListen (with setSunListen()) allows the correct interface to be displayed graphically, rendering the necessary component between `sunAwake` and `sunSleep` (*final situations* defined in 5.1).

5.2 Detailed the various cases

Always keeping the scheme shown in Figure 5.1 in mind, let's now examine all the possible cases more in detail. In order to analyze the functioning of the SoundRise application in more detail, four scenarios were defined:

- *scenario n.1 - startup of the application*
  i.e. when the application startup for the first time;

- *scenario n.2 - start listening*
  i.e. when the user clicks on the start button and the microphone turns on;

- *scenario n.3 - detected voice*
  i.e. when the microphone detects the voice;

- *scenario n.4 - stop listening*
  i.e. when the user click on the stop button and the microphone turns off.

5.2.1 Scenario n.1 - Startup of the application

This first scenario concerns the state in which the application is presented at the moment the application is first loaded into the device and every time the web page is reloaded and a user accesses it. The interface that appears to the user is the same as described in Section 5.1: a hilly landscape with a gradient blue sky is the landscape where the sun is collocated. It is in the center, on the lower part of the screen, and at the base of the hills there is the footer bar with the two buttons, start and stop.
At startup, no `audio context` of Web Audio API (chapter details in section 4.2) is created, and the *microphone is turned off*, so it does not capture any sound. So, the `useState` hooks presented before (subsection 5.1.1) are declared in such a way:

```
1 const [startButtonDisabled, setStartButtonDisabled] = useState(
    false);
2 const [stopButtonDisabled, setStopButtonDisabled] = useState(true);
3 const [isListening, setIsListening] = useState(false);
4 const [sunListen, setSunListen] = useState(false);
```

In fact, in the footer bar, the only button that can be used is the start button, which turns on the microphone. The other button, i.e. the *stop button is disabled*, therefore it cannot be pressed. Since the sunListen variable is set to `false`, the component that is shown is the `SunSleep` component.

The only operation that the user can do now is to press the start button, which is colored purple, in order to turn on the microphone. The first time this application will be loaded on a new device, as soon as the start button is clicked, it will ask for *permission* to access the microphone. Once the user accepts this permission, the application will start performing its operations. Furthermore, the device will no longer be asked to accept authorizations every time you press the start button: once accepted, the application saves this preference and remembers it for subsequent times.

5.2.2 Scenario n.2 - Start Listening

This scenario happens when the user clicks on the start button, in order to turn on the microphone. Once the start button is pressed, a series of operations and functions are called consequently. In particular, the handleStartListening() function set the start button as the disabled button, and the stop button became clickable. Furthermore, isListening will be set to `true`. While this variable still remains `true`, the sun is able to hear if it is able to recognize a voice sound. To do that, it is used the **useEffect** hook. The useEffect hook is a React hook that allows you to perform side effects in your functional components. Side effects can include data fetching, subscriptions, manual DOM manipulations, and more. It is a way to

introduce imperative logic into your React components, especially when you need to perform actions that happen outside the normal flow of React component lifecycles. In this case, the effect depends on the isListening state: the effect will be re-run if its value changes. The useEffect block sets up an audio processing environment when isListening becomes `true`. This includes creating an audio context, obtaining access to the user's microphone, and configuring various audio-related components like a dynamics compressor and an analyzer.

The most important function now is called initializeAudio. This function, which is called when isListening becomes `true`, is responsible for setting up the audio processing environment. The code of the initializeAudio() function is shown below:

```
1  const initializeAudio = () => {
2        audioContext = new (window.AudioContext || window.
            webkitAudioContext)();
3        navigator.mediaDevices
4          .getUserMedia({
5            audio: {
6              echoCancellation: true,
7              noiseSuppression: true,
8              autoGainControl: true,
9              highpassFilter: true,
10           },
11         })
12         .then((stream) => {
13           mediaStreamSource = audioContext.createMediaStreamSource(
                stream);
14
15           const compressor = audioContext.createDynamicsCompressor
                ();
16           compressor.threshold.value = -50;
17           compressor.knee.value = 40;
18           compressor.ratio.value = 12;
19           compressor.attack.value = 0;
20           compressor.release.value = 0.25;
21           compressor.reduction;
22
```

```
23            analyser = audioContext.createAnalyser();
24            analyser.fftSize = 2048;
25            mediaStreamSource.connect(analyser);
26            startListening();
27          })
28          .catch((err) => {
29            console.error(`${err.name}: ${err.message}`);
30          });
31      };
```

This code initializes and configures an audio processing setup using the Web Audio API. The first things to do to work with the Web Audio API are the creation and initialization of the `audio context`, which represents the audio processing graph and state. The constructor is defined with the *webkit* prefix so that it can work on various browsers.

Then the getUserMedia method is used to request access to the user's audio input device. Next, the createMediaStreamSource method is used to create a media stream source node from the obtained audio stream. A media stream typically represents a continuous flow of audio data. This is the media stream obtained from the user's audio input device using getUserMedia.
The method createMediaStreamSource is used to convert the incoming audio stream (from the user's microphone in this case) into a form that can be processed by the Web Audio API. With this source node (`mediaStreamSource`), it is possible to connect it to other nodes such as filters, compressors, or analyzers to manipulate or analyze audio data in real-time.
The specific parameters for compression and analysis are configured, including settings for echo cancellation, noise suppression, auto gain control, a high-pass filter, and a *DynamicsCompressorNode*, which applies dynamic range compression to the audio signal.

The next step is to create an `AnalyserNode`, used for frequency-domain analysis of the audio signal. The `fftSize` property is set to 2048, determining the size of the

Fast Fourier Transform (FFT)[1] used for the analysis. Then, it connects the media stream source to the analyzer, creating a connection between the audio input and the frequency analyzer.

The last important step is to call the startListening() function.
The startListening() function is responsible for processing the audio data obtained from the microphone and updating various visual and state elements based on that data. The function then retrieves data from the analyzer using getFloatTimeDomainData, storing it in the buf array, in this way:

```
1 analyser.getFloatTimeDomainData(buf);
2 const ac = setFrequency(buf, audioContext.sampleRate);
3 const vol = getStableVolume(buf, audioContext.sampleRate);
4 const v = getVowel(buf, audioContext.sampleRate);
```

getFloatTimeDomainData is a method provided by the Web Audio API, specifically for `AnalyserNode` instances. This method is used to obtain time-domain data of the audio signal being processed by the `AnalyserNode`.
The getFloatTimeDomainData method of the `AnalyserNode` retrieves the current time-domain data of the audio signal as a Float32Array. Analyzing the time-domain data allows applications to perform various audio processing tasks, such as pitch detection, volume analysis, and other real-time audio visualizations. In fact, it is used to calculate the pitch using the setFrequency function, the volume using the getStableVolume function, and the value related to the vowel sound using the getVowel function).

setFrequency() function implements the **Auto-Correlation Function (ACF2+)** algorithm to estimate the pitch frequency of an audio signal. The primary goal of the ACF2+ algorithm is to estimate the fundamental frequency (pitch) of a periodic signal by analyzing its auto-correlation function. The ACF2+ algorithm is a method for estimating pitch frequency in audio signals. It begins by preprocessing the in-

[1]The Fast Fourier Transform (FFT) is an important measurement method in the science of audio and acoustics measurement. It converts a signal into individual spectral components and thereby provides frequency information about the signal.

put signal, which may involve calculating the **Root Mean Square (RMS)**[2] and removing silent portions. The core step is computing the auto-correlation function, which identifies repeating patterns and their distance, which is related to pitch. To refine pitch estimation accuracy, the algorithm introduces steps like parabolic interpolation around detected peaks.
The final output is the estimated pitch frequency of the input audio signal, which is crucial for tasks like speech recognition, audio pitch correction, and musical instrument tuning.

The getStableVolume function calculates a volume value for an audio signal by first determining the sum of squares of its amplitudes and then computing the root mean square (RMS) value, which represents the average amplitude of the signal.
Subsequently, this RMS value is appended to an array termed `previousBuffers` to uphold a chronological record of past volume values, with the array size constrained to a predefined historical threshold.
Following this, the average volume is calculated by summating the values within the `previousBuffers` array and subsequently dividing by the array's length. The function returns the rounded average volume value, scaled by 100 for convenience or specific application requirements.
The purpose is to provide a stable representation of audio volume, considering both the current signal and its historical values to minimize amplitude variations over time.

The getVowel() function uses input signal and sample rate to extract information about vowel characteristics. The function begins by setting the input signal and sample rate, subsequently initiating a sequence of operations to extract relevant information about the vowel characteristics. The steps include autocorrelation computation, linear predictive coding (LPC) analysis, and formant extraction. Initially,

[2]The root mean square (RMS) is the square root of the mean square, which is the arithmetic mean of the squares of a group of values. One of the principal applications of RMS values is with alternating currents and voltages. The value of an AC voltage is continually changing from zero up to the positive peak, through zero to the negative peak and back to zero again. The RMS value is the effective value of a varying voltage or current.

the autocorrelation function is calculated using an efficient implementation with a pre-computed signal.
Subsequently, LPC analysis is performed through the Durbin algorithm, yielding a set of linear predictive coding coefficients (LPC). These coefficients are then utilized to obtain the roots of the polynomial using the Durand-Kerner method. The obtained roots are processed to identify valid formants and specific frequency and bandwidth values are extracted. These formants are then compared against predefined ranges associated with distinct vowel sounds.
The function returns an array of formants, and the recognized vowel sound is logged to the console. It is noteworthy that the accuracy of vowel recognition is contingent on the correlation of formant frequencies with predefined ranges, and adjustments to these ranges may enhance or modify recognition accuracy.
The overall functionality of the getVowel() function contributes to the domain of speech analysis and phonetics, facilitating the identification of vowel sounds in an audio signal.

Basically, every time the startListening() function is called, these three functions return values that the startListening() function compares with precise parameters (which will be discussed in the next subsection). This comparison is used to determine whether the recognized sound is the sound of a voice and therefore should be shown graphically. Until this happens, i.e. until the sound is the sound of a voice and/or the values returned by the three functions do not pass the comparison with the default values, the sunListen variable will always remain set to `false`, so the component shown is still the `SunSleep` component.

5.2.3 Scenario n.3 - Detected Voice

Based on the theory studied (section 4.1), the values of the pitch and volume range have been defined, in such a way as to be able to determine when the perceived sound is actually the sound of a voice.

As regards the pitch, it was decided to use the range from 150 to 600 Hz. Instead, as regards the volume it was decided to keep a fairly large scale, considering that children (especially if deaf) have more tendency to have a high-pitched voice. The range for volume is [2, 30]. For vowels, it is simpler: if the vowel is recognized we have a value to show, otherwise not. Clearly, if even just one of the pitch or volume values is outside the decided ranges, the value of the vowel is not valorized: in fact, in order for an acceptable result to be shown, all the values must be valid (vowels, pitch, and volume).

When three acceptable values are obtained, based on the size of the screen in which the application is executed, a Javascript library called `setSunFeature` defines the values for `yCoordValue` and `radValue`, respectively for the position in the vertical axis of the pitch and for the size of the radius of the sun for the volume.
To avoid a sudden change in graphic feedback (perhaps due to sudden noises or the attack), two buffer arrays were created, one for the pitch and one for the volume, which retains the last n values (in this case $n = 600$). The result shown graphically is the average of the values contained in that array. Only acceptable values are retained in this array, and in addition, after 50 frames of no voice sound, this array is completely cleaned.

Furthermore, the relevant color for the corresponding vowel is assigned. By default the sun is yellow, and considering a study [55] in which it is shown that timbre and color are closely related in visual perception, with significant associations between letters and colors observed in both synesthetic and non-synesthetic subjects, it was decided to represent the associations between the vowel and colors as it follows:

Vowel	**Color**
A	Red
E	Green
I	Blue
O	Orange
U	Gray

In Figure 5.2 it is possible to see some screenshots taken while the application was running. Each of them shows a different vowel, understandable by the different colors of the sun, as follows:

- In Figure 5.2 (a), the sun is red, so the vowel shown is A.
  The pitch value obtained is 425Hz, while the intensity value is 9.

- In Figure 5.2 (b), the vowel E is represented, as the sun is green.
  The pitch value obtained is lower than in the previous figure (precisely 296Hz) as is the intensity value, which in this case is 4.

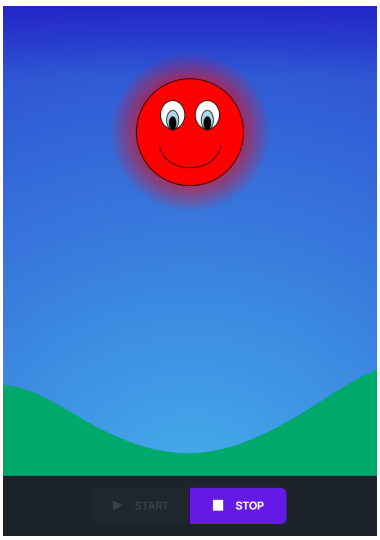
- Figure 5.2 (c) represents the vowel I, as the sun is blue.
  The detected pitch value is 277Hz, while the intensity value is now slightly higher, precisely it is 7.

- In Figure 5.2 (d), the sun is orange, so the vowel represented is O.
  The pitch value obtained is 307Hz, while the intensity value this time is larger than in the previous cases, as it is 14.

- In Figure 5.2 (e), the sun is grey, since the vowel represented is U.
  Here, the detected pitch value is 359Hz, while the intensity value is 11.

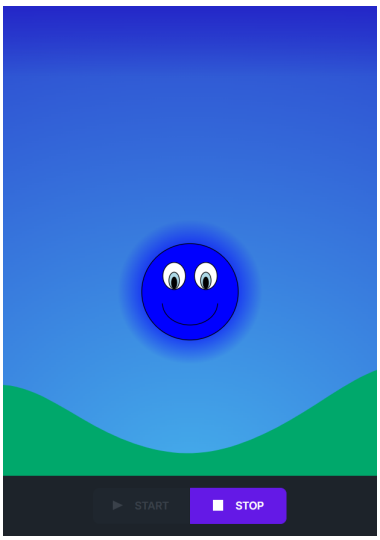
After all these operations, the sunListen state variable will be set to true, so now the component shown is the `SunAwake` component.

In summary, when the microphone captures sounds that are recognized as speech, it calculates the characteristics of the sun based on the pitch, volume, and vowel values in relation to the screen size and shows them graphically with the `SunAwake` component, while when it does not recognize any sound by voice, uses the characteristics of the sun based on the size of the screen and, placing the sun in the center and at the lowest point of the screen, displays them graphically with the `SunSleep` component.

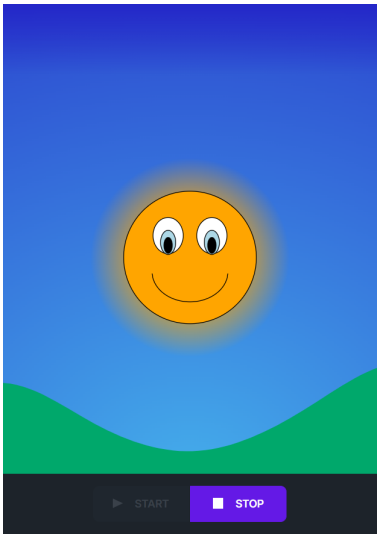


(a)
Vowel represented: A
Pitch detected: 425Hz
Sound intensity value: 9

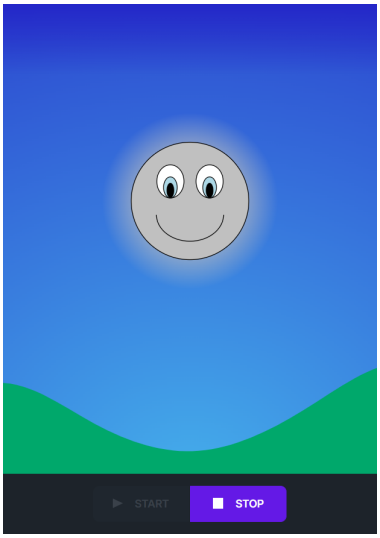
(b)
Vowel represented: E
Pitch detected: 296Hz
Sound intensity value: 4



(c)
Vowel represented: I
Pitch detected: 277Hz
Sound intensity value: 7



(d)
Vowel represented: O
Pitch detected: 307Hz
Sound intensity value: 14



(e)
Vowel represented: U
Pitch detected: 359Hz
Sound intensity value: 11

Figure 5.2: Some screenshots taken while running the application showing the sun with its different colours, depending on the vowel recognised:
(a) Representation of vowel A **(b)** Representation of vowel E
(c) Representation of vowel I **(d)** Representation of vowel O
(e) Representation of vowel U

5.2.4 Scenario n.4 - Stop Listening

When the user decides to turn off the microphone he must click on the stop button. Consequently, the button situation returns to being like that when the application is started: stopButtonDisabled is set to true, while startButtonDisabled, isListening, and sunListen are set to false. With the `SunSleep` component, the sleeping sun is placed in the center and at the lowest point of the screen.

To stop the microphone stream, it is necessary to do a few more steps with the Web Audio API. In fact, the stopListening() function serves the purpose of halting the real-time audio processing initiated by the startListening() function: it encompasses a series of operations to reset and disconnect audio-related components, effectively terminating the audio capture and analysis process.
After cleaning all previously created buffer arrays, it is responsible for ending the flow of audio data and closing the audio processing context.
The process of disconnecting audio components involves stopping the incoming audio stream (`mediaStreamSource` disconnection) and concluding the audio processing session (`audioContext` closure). Closing the audio context signals the conclusion of the audio processing session, terminating ongoing audio operations and releasing associated resources. Upon successful closure of the audio context, the variables linked to audio components (`audioContext`, `analyzer`, and `mediaStreamSource`) are explicitly set to null. This action clears references to these objects, indicating that they are no longer in use and allowing for proper resource cleanup. This piece of the stopListening() function is written as follows:

```
1 mediaStreamSource.disconnect();
2 audioContext
3     .close()
4     .then(() => {
5         audioContext = null;
6         analyser = null;
7         mediaStreamSource = null;
8     })
```

6 Conclusions

In the pursuit of advancing speech therapy solutions for individuals facing communication difficulties, the development of SoundRise 2.0 marks an interesting step forward. After years of research and prototypes at the CSC (Centro di Sonologia Computazionale), this application stands as a testament to the successful implementation of a meticulous reactivation process.

The reactivation process, carefully designed and executed, has not only breathed new life into previously shelved prototypes but has led to the creation of a functional and impactful tool.
SoundRise 2.0, built on React JS and leveraging the Web Audio API, provides an intuitive platform for users, especially children, to independently practice speech therapy exercises. The animated sun interface, mirroring tonal and timbre characteristics, offers real-time visual feedback, creating a dynamic and engaging experience for users.

The application's success is not only measured by its functionality but also by its potential to enhance the communication capabilities and overall quality of life for its users. The approach taken in the reactivation process has proven to be not only correct but transformative, resulting in a maintainable and multiplatform application that is ready for use.

While SoundRise 2.0 is a culmination of the present, it also serves as a foundation for the future. Thanks to the reactivation process, the deserved longevity of the application is finally guaranteed. This not only secures the investment in time and

effort put into its development but also ensures that SoundRise 2.0 will continue to be a reliable resource for those seeking to enhance their communication skills. As technology evolves and greater and continuously deepens the understanding of the world of speech therapy (especially for children), the potential for enhancements to further improve user experience remains promising. Continuous refinement in both function and code will undoubtedly contribute to the longevity and impact of SoundRise 2.0.

The ongoing contribution to and improvement of the code, exemplified by the open-source nature of GitHub where all the versions of the SoundRise application have been archived, adds a layer of transparency and collaboration to the project. In this way, the code is now open for scrutiny and enhancement by a wider community of developers. SoundRise's open-source model fosters innovation and problem-solving by involving developers from diverse backgrounds and expertise. This approach creates a dynamic ecosystem where ideas flow freely, leading to continuous improvements and innovative solutions. Open-source development instills trust and confidence among users, allowing them to inspect and verify code quality and integrity. This transparency empowers the user community and holds developers accountable for the quality of the code. The open-source ethos encourages the sharing of knowledge and best practices, allowing developers to learn from each other and gain exposure to different coding styles and techniques. This knowledge exchange contributes to professional growth and overall improvement in the software development community.

Considering future developments and improvements, it is possible to imagine the SoundRise application as a versatile tool, which is capable of expanding its usefulness beyond speech therapy.

For example, with its animated sun interface and real-time feedback capabilities, the application could serve as a valuable support tool for children's singing lessons, particularly in the context of preparatory music education. The dynamic nature of SoundRise 2.0 not only facilitates speech therapy exercises but also opens new

possibilities for fostering musical skills in young learners. By engaging with the visual representation of their voice, children can learn to recognize the pitch and intensity of their vocalizations. This focus on sound nuances is crucial for developing an understanding of diaphragmatic breathing, a fundamental skill in both speech therapy and musical training. SoundRise 2.0, therefore, becomes not just a tool for communication improvement but a holistic platform that nurtures vocal and respiratory skills essential for comprehensive development.

Looking still forward, to transforming SoundRise 2.0 into a full-fledged digital therapy, it is possible to envise a variant that *gamifies* the experience. The ultimate goal from the inception of SoundRise has been to pioneer game-based learning in the realm of speech therapy. A further upgrade could be given by restructuring the application into a level game. For example, the first level could tell the user that to win and reach the next level, the sun must turn green for more than 5 seconds in a certain position on the screen (pitch) and with a certain size (volume). The next level would then do the same thing perhaps with the same pitch and volume values, but with a different vowel. The next one instead modifies the pitch value. And so on.

So, introducing a level-based game approach not only aligns with this fundamental vision but also holds the promise of revolutionizing the therapeutic side. By infusing a gamified structure, we can elevate user engagement, especially for our younger audience, by seamlessly integrating therapeutic exercises into an interactive and enjoyable challenge. This transformative approach not only turns SoundRise 2.0 into a tool for communication improvement but propels it into the realm of edutainment, where learning becomes an adventure, and every achievement is a step toward enhanced communication skills and overall well-being.

Note how this development could also be very useful for the singing lessons mentioned above because it would stimulate even more the work of pitch correction, sound intensity, and breathing control (to maintain the duration of the sound for a certain value of seconds). As regards the improvement of the code, the current

method employed for vowel detection within SoundRise 2.0 could be refined. The existing code identifies the five vowels in the Italian language extracting the relative formants.
However, this approach raises concerns about scalability and inclusivity. Customizing formants for different demographics risks perpetuating discriminatory biases, undermining the robustness we uphold in the reactivation process [56].
To resolve this, an idea is to explore the integration with the `VowelWorm` project [57]. Developed at the Department of Computational Perception at Johannes Kepler Universität Linz, the `VowelWorm` projects used mathematical models with low computational cost which makes it possible to predict and display vowel quality in real-time, and this approach is based on common acoustic features (mainly MFCCs[1]) [58]. It leverages machine learning algorithms, like linear classification and regression, to project spoken or sung vowels into a continuous articulatory space: the IPA vowel graph, which shows the evolution of sung vowels over time in an intuitive way. The principles embedded in the `VowelWorm` project align with the commitment to fostering inclusivity and avoiding the demographic bias that SoundRise proposes. So, the concepts in this work can be analyzed in depth to find a connection with the SoundRise application, to be able to integrate and improve together.

This not only fosters innovation but also positions SoundRise 2.0 as a living project, capable of adapting to emerging technologies and evolving user needs. Furthermore, the sustainability of research and development cannot be understated. As we celebrate the success of SoundRise 2.0, it serves as a reminder to embrace a sustainable approach to technology. Rather than a cycle of production and disposal, let our efforts be a beacon for a future where engineering endeavors contribute to lasting solutions and enduring impact.

[1]MFCCs (Mel Frequency Cepstral Coefficients) are a compact representation of the spectrum of an audio signal. MFCCs are used to represent the spectral characteristics of sound in a way that is well-suited for various machine learning tasks, such as speech recognition and music analysis.

In conclusion, the process-driven passage from the conceptualization to the realization of the SoundRise 2.0 application has been a success. SoundRise 2.0 not only reflects the CSC's dedication to promoting technological improvement for learning activities but also stands as a beacon of hope for individuals seeking to overcome communication challenges. The future path includes not only the continuous improvement of SoundRise 2.0 but also the exploration of new horizons in speech technology.

References

[1] British Deaf Association. *British Deaf Association - Definitions of hearing impairments*. URL: https://www.derbyshire.gov.uk/site-elements/documents/pdf/social-health/adult-care-and-wellbeing/disability-support/hearing-impaired/british-deaf-association-definitions-of-hearing-impairments.pdf.

[2] ConnectHear.org. *The difference between D/deaf, hard of hearing and hearing-impaired*. 2020. URL: https://www.connecthear.org/post/the-difference-between-d-deaf-hard-of-hearing-and-hearing-impaired.

[3] NAD - National Association of the Deaf. *Frequently Asked Questions*. URL: https://www.nad.org/resources/american-sign-language/community-and-culture-frequently-asked-questions/.

[4] J.R. Gannon. "Deaf Heritage: A Narrative History of Deaf America". In: *Deaf Heritage: A Narrative History of Deaf America* (Jan. 2011), pp. 1–483.

[5] Ente Nazionale Sordi ETS-APS. *Lingua dei Segni Italiana*. URL: https://www.ens.it/lingua-dei-segni/.

[6] Murphy J. J. in Encyclopedia Britannica. *Demosthenes*. URL: https://www.britannica.com/biography/Demosthenes-Greek-statesman-and-orator.

[7] Royal College of Speech and Language Therapists. *RCSLT and speech and language therapy history*. URL: https://www.rcslt.org/about-us/history/.

[8] Judith Felson Duchan. *A History of Speech - Language Pathology*. URL: https://www.acsu.buffalo.edu/~duchan/new_history/overview.html.

[9] De Cagno G. Mollo F. Citro R. Roch M. "History of SLT (Speech and Language Therapists) and developmental language disorders in Italy". In: May 2019.

[10] Bluder T. Eikerling M. Rinker T. Lorusso M. L. "Speech and Language Therapy Service for Multilingual Children: Attitudes and Approaches across Four European Countries". In: (2021). URL: https://www.mdpi.com/2071-1050/13/21/12143.

[11] Colletti L. Nicastri M. *La terapia uditivo-verbale (AVT) nell'esperienza italiana*. Jan. 2021. URL: https://rivistedigitali.erickson.it/logopedia/archivio/vol-7-n-1-2/la-terapia-uditivo-verbale-avt-nellesperienza-italiana/.

[12] Chiarelli V. *Il metodo bimodale in terapia logopedica per i bambini sordi*. Jan. 2021. URL: https://www.centromedicoriabilitativo.it/blog/2021/01/metodo-bimodale/.

[13] Kiversal Blog. *Prelingual and postlingual hearing loss: communication problems*. Feb. 2019. URL: https://blog.kiversal.com/en/prelingual-hearing-loss/.

[14] American Speech-Language-Hearing Association. "Effects of Hearing Loss on Development". In: (2005). URL: https://www.readingrockets.org/topics/speech-language-and-hearing/articles/effects-hearing-loss-development.

[15] NAD - National Association of the Deaf. *Educational Placements*. URL: https://www.nad.org/resources/education/k-12-education/educational-placements/.

[16] Calderon R. Greenberg M. "Social and Emotional Development of Deaf Children: Family, School, and Program Effects". In: *The Oxford Handbook of Deaf Studies, Language, and Education: Second Edition* (Sept. 2012).

[17] Anita Cloete. "Technology and education: Challenges and opportunities". In: *HTS Teologiese Studies / Theological Studies* 73 (Oct. 2017).

[18] School of Education Online - American University. *How Important Is Technology in Education? Benefits, Challenges, and Impact on Students*. June 2020. URL: https://soeonline.american.edu/blog/technology-in-education/.

[19] Michael Stinson. "Current and Future Technologies in the Education of Deaf Students". In: *The Oxford Handbook of Deaf Studies, Language, and Education* (Jan. 2012).

[20] Omose I. *The Impact of Technology on Deaf Culture: How Technology is Shaping the Future of Deaf Identity*. Feb. 2023. URL: https://www.unspokenasl.com/aslblogs/the-impact-of-technology-on-deaf-culture-how-technology-is-shaping-the-future-of-deaf-identity/.

[21] Di Franco M. *L'arte del gioco e il suo valore educativo*. URL: https://site.unibo.it/griseldaonline/it/didattica/marcella-di-franco-arte-gioco-valore-educativo.

[22] Save the Children Blog e Notizie. *GAME BASED LEARNING, GAMIFICATION E DIDATTICA: COSA SONO*. Apr. 2020. URL: https://www.savethechildren.it/blog-notizie/game-based-learning-gamification-e-didattica-cosa-sono.

[23] Tamosevicius R. *Why Is Game-Based Learning Important?* Nov. 2022. URL: https://elearningindustry.com/why-is-game-based-learning-important.

[24] Fastelli A. "Implicit Learning and Deafness: from the Assessment to the Design and Implementation of a Serious Game-based Training for Deaf Children with Cochlear Implants". PhD thesis. University of Padua, Dec. 2019.

[25] Bycer J. in game-wisdom.com. *THE IMPORTANCE OF FEEDBACK TO LEARNING IN GAME DESIGN*. Mar. 2015. URL: https://game-wisdom.com/critical/feedback-game-design.

[26] J. Macauslan H. Fell C. Cress. "VisiBabble for reinforcement of early vocalization". In: *ACM Sigaccess Accessibility and Computing* (2004), pp. 161–168.

[27] Z. Kacic K. Vicsi P. Roach. “SPECO - a multimedia multilingual teaching and training system for speech handicapped children”. In: Sept. 1999, pp. 859–862.

[28] Articulation Teacher. *Teach Speech At Home!* URL: https://www.articulationteacher.com/.

[29] Little Bee Speech. *Professional Software for Speech,Language and Literacy.* URL: https://littlebeespeech.com/.

[30] Hanks H. *Mommy Speech Therapy - Thoughts on speech and language development.* URL: https://mommyspeechtherapy.com/.

[31] Aardman. *StorySign.* URL: https://www.aardman.com/interactive/storysign/.

[32] Ezekiel S. and Google Creative Lab. *Look to Speak.* Dec. 2020. URL: https://experiments.withgoogle.com/looktospeak.

[33] Giusto S. Rodà A. “Soundrise: Studio e Progettazione Di un’Applicazione Multimodale Interattiva Per La Didattica Basata sull’Analisi Di Feature Vocali”. MA thesis. University of Padua, July 2012.

[34] Randon M. Avanzini F. “Soundrise: Sviluppo E Validazione Di un’Applicazione Multimodale Interattiva Per La Didattica Basata Sull’analisi Di Feature Vocali”. MA thesis. University of Padua, July 2012.

[35] *Pure Data Documentation.* URL: https://puredata.info/.

[36] Brent W. “A Timbre Analysis And Classification Toolkit For Pure Data”. In: (2010), pp. 2–7.

[37] Turetta G. Canazza Targon S. Fiordelmondo A. “Soundrise 2.0: Sviluppo di un’interfaccia grafica interattiva in three.js per supportare persone con disabilità uditive”. MA thesis. University of Padua, Mar. 2023.

[38] *Three.js documentation.* URL: https://threejs.org/docs/.

[39] *Blender documentation.* URL: https://docs.blender.org/.

[40] Fila R. Canazza Targon S. Fiordelmondo A. “SoundRise 2.0: Sviluppo di un modello di riconoscimento timbrico per un sistema di assistenza web dedicato a persone con disabilità uditive”. MA thesis. University of Padua, Sept. 2023.

[41] Fiordelmondo A. Russo A. Pizzato M. Zecchinato L. Canazza S. “A multi-level dynamic model for documenting, reactivating and preserving interactive multimedia art”. In: *Frontiers in Signal Processing* 3 (2023). URL: https://www.frontiersin.org/articles/10.3389/frsip.2023.1183294.

[42] Canazza S. De Poli G. Vidolin A. “Gesture, Music and Computer: The Centro di Sonologia Computazionale at Padova University, a 50-Year History”. In: *Sensors* 22 (2022). URL: https://www.mdpi.com/1424-8220/22/9/3465.

[43] Bressan F. Canazza S. “The challenge of preserving interactive sound art: a multi-level approach”. In: *International Journal of Arts and Technology (IJART)* 7 (Nov. 2014).

[44] Depocas A. “Documenting and conserving technological art: the evolution of approaches and methods”. In: *Digital Art Conservation: Theory and Practice, Ambra, Karlsruhe* (2013), pp. 145–153.

[45] The National Archives. *A2A: Proposed Cataloguing Standards for the National Archives Network*. URL: https://cdn.nationalarchives.gov.uk/documents/cataloguing_standards.pdf.

[46] Bressan F. Canazza Targon S. Rodà A. Orio N. “Preserving today for tomorrow: A case study of an archive of Interactive Music Installations”. In: (2009).

[47] *OpenGL*. URL: https://www.opengl.org/.

[48] *Responsive Viewer*. URL: https://responsiveviewer.org/.

[49] MacCallum J. in MUsic Technology Online Repository. *Unit 6: Timbre*. URL: https://mutor-2.github.io/MUTOR/units/06.html.

[50] Stanley R. Alten. *Audio in Media*. Ninth. Wadsworth - Cengage Learning, Jan. 2010.

[51] Smus B. *Web Audio API*. O’Reilly, 2013.

[52] *Le Web Audio API di HTML5*. URL: https://www.mrw.it/html/web-audio-api-html5_12070.html.

[53] Leonard S. Turner W. *JavaScript for Sound Artists - Learn to Code with the Web Audio API*. CRC Press, 2013.

[54] *React Documentation*. URL: `https://react.dev/learn`.

[55] Julia Simner et al. "Non-random associations of graphemes to colours in synaesthetic and non-synaesthetic populations". In: *Cognitive neuropsychology* 22.8 (2005), pp. 1069–1085.

[56] *Teaching and Learning with Technology at Reed: the Vowel Worm*. URL: `https://blogs.reed.edu/ed-tech/2015/03/the-vowel-worm/`.

[57] Harald Frostel, Andreas Arzt, and Gerhard Widmer. "The vowel worm: Real-time mapping and visualisation of sung vowels in music". In: *Proceedings of the 8th Sound and Music Computing Conference*. Citeseer. 2011, pp. 214–219.

[58] *VowelWorm GitHub Repository*. URL: `https://byu-odh.github.io/apeworm/`.