Machine Audition Curriculum and Real-Time Music Accompaniment

by

Nada Hussein

B.S., Massachusetts Institute of Technology (2020)

Submitted to the Department of Electrical Engineering and Computer Science
in partial fulfillment of the requirements for the degree of

Master of Engineering in Electrical Engineering and Computer Science

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

September 2021

Author .....
Department of Electrical Engineering and Computer Science
August 6, 2021

Certified by .....
Cynthia Breazeal
Professor of Media Arts and Sciences
Thesis Supervisor

Accepted by .....
Katrina LaCurts
Chair, Master of Engineering Thesis Committee

Machine Audition Curriculum and Real-Time Music Accompaniment

by

Nada Hussein

Submitted to the Department of Electrical Engineering and Computer Science
on August 6, 2021, in partial fulfillment of the
requirements for the degree of
Master of Engineering in Electrical Engineering and Computer Science

Abstract

A machine audition curriculum was created as part of the MIT Media Lab's Artificial Intelligence Education initiative. This curriculum was geared towards middle school students to help them understand how humans and machines perceive sound, and allow them to apply this knowledge to create and analyze their own music. This thesis presents the tools created to aid in the teaching of this curriculum: a new music audition Scratch extension. This extension introduces the ability to create and analyze music, as well as the integration of Google Magenta, a machine learning library that allows students to generate new music or accompany music that they have created. Through the use of this Scratch extension, it was possible to pilot the machine audition curriculum with middle school students and show that they were able to better understand signal properties, create and analyze their own music, and understand the similarities and differences between human and machine audition.

Thesis Supervisor: Cynthia Breazeal
Title: Professor of Media Arts and Sciences

Acknowledgments

First, I would like to thank Stephen Kaputsos, who worked closely with me throughout this entire project, and was there through all of the middle school pilots, successful or not. Thank you for the time and energy you poured into this project with me to get it to where it is today.

I would also like to thank my thesis advisor, Cynthia Breazeal, for supporting my project throughout the process, and always encouraging me to demo my contributions as soon as possible. Our conversations provided me with valuable feedback and ideas to improve my Scratch blocks, both of which helped me create a project that I can be truly proud of.

I would also like to thank the entire Personal Robots Group at the Media Lab for supporting me throughout this project. Hearing from your own experiences with AI education, and being able to rely on your support during a fully remote project in COVID was truly what made this project possible. Thank you for always knowing where I could find Scratch documentation, offering to read over my materials and give me feedback, and hop on Zoom calls with me to learn more about my project and give me ideas for improvement. The PRG community is a significant reason why this experience has been so positive for me.

I would also like to thank Christina Kaputsos, who designed all of our visual assets for the curriculum. Our story characters, Lucas and Audii, their engaging backstories, as well as the beautiful icons for the curriculum and the Scratch extension would not exist if not for you.

I also want to thank the Advanced Math and Science Academy Charter School, and specifically Padmaja Bandaru, for making the time to pilot the machine audition curriculum with us during your class time. It was a pleasure working with students on the curriculum, and seeing student interface with the tools we had created. The feedback and insight gained from this pilot was also invaluable to my work, and helped me gain a deeper understanding of the needs of middle school students in order to provide an effective learning experience. Thank you, Padmaja, for bringing

your educational experience and understanding of your students to our pilots, and making them much more effective than they could have been otherwise.

I would like to thank my friends for always supporting me. Thank you for the spontaneous boba runs, the late night virtual company when I was exhausted but still had more to do, and the constant emotional support throughout a challenging year.

Finally, I would like to thank my parents, Iman Salama and Ahmed Hussein, for supporting me for years in my education and career. Thank you for always being excited to talk to me about my work and whatever I happen to currently be learning about, and giving me the opportunity and encouragement to discover and act on my interests. I would not be where I am without you.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

As part of an Artificial Intelligence Education initiative throughout the Media Lab, the Personal Robots group has developed a machine audition curriculum aimed at middle school students. The curriculum focuses on the mechanics of human and computer hearing, comparing how the human ear operates to the methods in which computers perceive and respond to audio through signal processing and machine learning. The curriculum aims to educate students on machine audition while engaging them in creative learning and computational thinking, such that they are able to apply their learning to music projects of their own. However, it is important to ensure that students are able to use a computational platform that makes it easy for them to implement new ideas. A visual programming language called Scratch [5] aims to fill this need, allowing younger students to get hands-on programming experience without the complexity of a non-visual programming language. Thus, Scratch functionality must be expanded such that the machine audition curriculum can be taught with Scratch as a tool to help students engage in interactive activities meant to solidify their understanding of signal processing, music concepts, and human-AI collaboration in music projects. To implement and pilot a successful machine audition curriculum, the following overlapping goals were created:

1. Create a Scratch extension for use in a Machine Audition Curriculum that can support signal analysis and visualization, and music creation and accompani-

ment.

2. Interface with Machine Audition Curriculum to create interactive, educational activities that can be done using the Scratch blocks created.

3. Support middle-school students in learning about machine audition.

In this paper, I describe the machine audition curriculum, and explore the implementation of Scratch blocks created to facilitate machine audition education in middle school students, as well as the results that were found through pilots with the students.

1.1 Machine Audition Curriculum Overview

The machine audition curriculum, designed in collaboration with other Personal Robot Group members, guides middle school students through the mechanics of both human and machine hearing. The curriculum covers 5 main modules – audition overview, signals, sensation, perception, and human-AI collaboration. With these modules, students learn about the entire hearing and perception pipeline that both humans and machines use to understand the sounds around them. The curriculum balances lecture format modules with discussion questions and activities designed to engage students and solidify their learning by giving them the opportunity to talk through the concepts. The curriculum closes with a larger-scale machine audition capstone project, allowing student to utilize all the tools they have been introduced to thus far in order to create their own creative musical project that combines signals understanding with machine learning tools.

1.2 Why Use Scratch?

Scratch [5] is a block-based, visual programming language that has been used in a variety of contexts to help students learn how to program. It allows students to abstract away the complex syntax of code while learning about programming logic. Given the

middle school target audience, it was concluded that it is not feasible to construct hands-on activities in a standard programming language since the students do not have any programming knowledge. However, it is still important to allow students to have experience with the computational element of music creation, and use these tools to visualize what they are learning and solidify their understanding with open-ended, interactive projects. Scratch allows younger students to create functional, creative programs that can facilitate the learning we were hoping to provide through this curriculum. Thus, it was determined that Scratch was the best fit in this context for this curriculum, as it would improve the learning curve for younger students, and still enable them to experience the computational elements of music analysis, creation, and accompaniment. Throughout this project, a new Scratch extension was built that allowed students to create and analyze their own music, as well as utilize machine learning in order to create more complex and interesting music to accompany their own.

1.3 Outline

Chapter 2 explores the current works that contribute to the development of this project. Chapter 3 describes the full machine audition curriculum content that is presented to students. In Chapter 4, research and design choices that were made in creating the Scratch blocks are presented, as well as the activities that were designed to be implemented during the curriculum using the created Scratch blocks. In Chapter 5, the architecture of the Scratch extension is shared, as well as all of the functionality it has and each block's use case. This is followed in Chapter 6 with an evaluation of both the curriculum and the Scratch blocks, after piloting the lessons with middle-school students in Spring 2021. In Chapter 7, future improvements are discussed that could be made on the Scratch blocks in order to improve their usability and educational ability in later iterations of the curriculum and as a free-standing extension. Finally, in Chapter 8, overall contributions of the project are discussed, with focus on its current achievements and limitations.

Chapter 2

Related Works

This section shares existing Digital Signal Processing (DSP) libraries, machine learning libraries, and Scratch functionality that was researched to find the gaps that needed to be filled in order to give students the tools to interact with the Machine Audition curriculum. A similar AI educational initiative is also introduced, which the machine audition curriculum was based on.

2.1 Sight of AI Curriculum

The MIT Media Lab has designed other curricula as part of an AI education initiative. One of these is the Sight of AI curriculum, which focuses on human and computer vision. The Sight of AI curriculum teaches middle school students about the impact of vision on humans' cognitive abilities, and compares the human eye to the processes machines use to see through cameras. The students are then able to apply their knowledge by creating image-driven Scratch games. This curriculum also includes the design of two characters meant to drive the storyline behind the curriculum – Percy, and a robot companion named Vizzi, who features a camera-like design as seen in figure 2-1.

This curriculum became the template that was followed for the Sound of AI machine audition curriculum.

Figure 2-1: Sight of AI: Percy and Vizzi.

2.2 Digital Signal Processing

One of the most widely used DSP libraries is scipy.signal [4], a Python library that is able to perform standard DSP functions, such as Fast Fourier Transforms, Short Time Fourier Transforms, and filtering in order to extract certain frequencies from signals. This library is able to compute these transformations quickly, and visualize them in intuitive ways. A visual of scipy.signal's standard Fourier Transform as well as spectrogram visualizations can be seen in Figures 2-2 and 2-3.

This library represents an optimal signal processing backend – however, it is not compatible with Scratch and JavaScript. To implement a DSP library in Scratch, one would need to implement a new library, as there are currently no DSP libraries that are supported by Scratch.

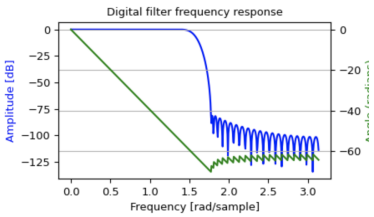


Figure 2-2: scipy.signal Fast Fourier Transform.

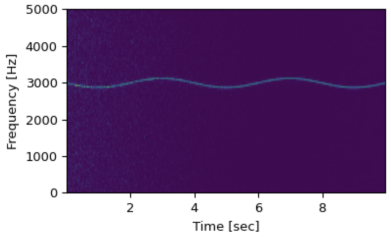


Figure 2-3: scipy.signal Spectrogram.

2.3 Machine Learning

There are plenty of existing machine learning libraries, including Tensorflow [6] for JavaScript and PyTorch [3] for Python. Google Magenta [1] is a TensorFlow-powered machine learning library that can be used in both Python and JavaScript. It focuses on facilitating creative generation of art and music, and is able to generate music on command with a machine learning backend. It is able to change the music it generates given a user's inputs, such as temperature (how fast-paced the resulting music should be) and the duration of music. Magenta is also able to save and visualize the music it creates in intuitive ways. Google Magenta offers all of the music generation and accompaniment functionality that is needed in the machine audition curriculum, so it is an ideal candidate for the Scratch backend.

2.4 Prior Scratch Functionality

2.4.1 Music Visualization

Scratch is currently unable to support plotting visualizations, or text rendering to label graphs. It is able to render text as speech bubbles attached to a sprite, but can not rotate text, change sizing, or put text in a location separate from a sprite. An example of Scratch's current text rendering can be seen in Figure 2-4.

Furthermore, there is no plotting backend available in Scratch. The only extension available for writing is the Pen extension, which allows users to set pen colors, and

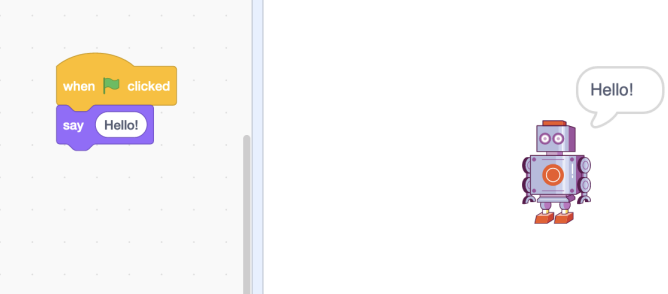


Figure 2-4: Current Scratch Text Rendering Functionality.

give a command to position the pen up or down. Users control the pen's writing by using the character movement blocks, specifying a direction and distance to move, and the pen draws the path the character followed. An example of this can be seen in Figure 2-5.

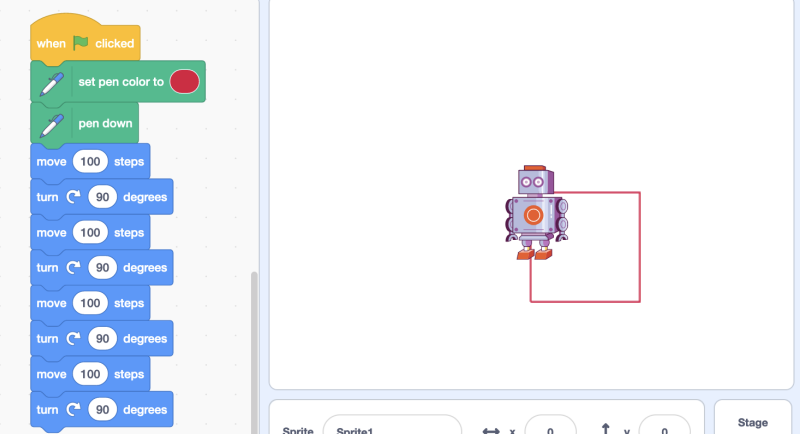


Figure 2-5: Current Scratch Plotting Functionality.

Both the Pen and Motion extensions in Scratch offer suitable functionality that can provide a robust backend for a plotting and text rendering library for music visualization.

2.4.2 Music Creation

Scratch currently does have a Music extension. This allows users to select an instrument, and play notes by specifying a key on a piano, as well as a duration in seconds. Users can also change tempo, and include rests in their music. A visual of the Music blocks, along with the virtual piano note selection tool, can be seen in Figure 2-6.

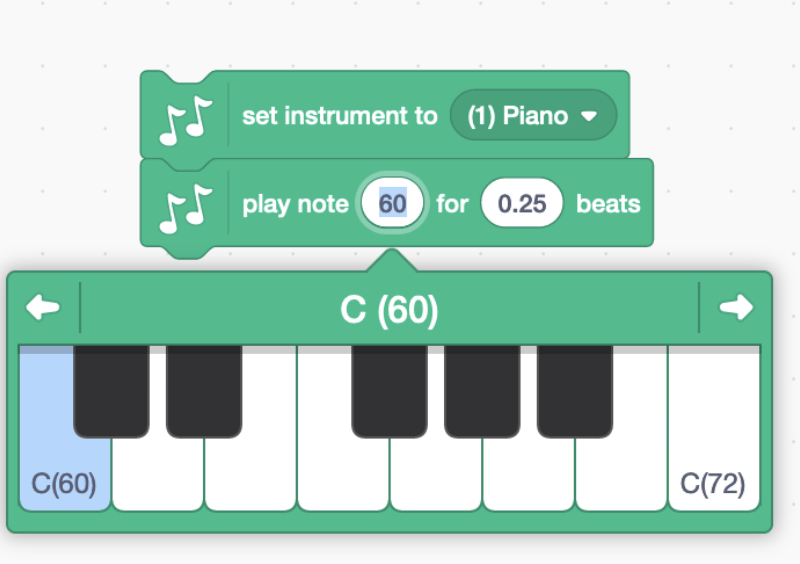


Figure 2-6: Current Scratch Music Creation Functionality.

This extension does not support changes in volume, and does not have a robust method of storing and modifying the music. This means the code does not retain any past information about instrument or note selection, making it impossible to build music analysis and visualization tools off of the current implementation.

2.4.3 Music Accompaniment

Scratch does not provide any machine learning backend for music accompaniment. It does, however, offer some extensions that interface with more general machine learning libraries. Namely, Dancing with AI PoseBlocks [8] use a machine learning backend and the user's camera to allow users to create movement-based AI systems. The backend implementation of these blocks, and the way they interface with a machine learning library, demonstrates a reliable way of integrating machine learning and general external libraries with Scratch.

Chapter 3

Machine Audition Curriculum

The basis of this project is the machine audition curriculum, designed and created in collaboration with the Personal Robots Group. The curriculum is aimed at middle school students and covers both human and computer perception of audio, ending with a larger machine audition capstone designed to help students use all of their new knowledge to create an exciting, human-AI collaborative music project in Scratch. Each of the 5 modules is expanded upon further in this chapter.

3.1 Module 1: Audition Overview

In the first module of the curriculum, middle school students are introduced to the concept of audition, or hearing. The goal is to allow them to understand how humans' sense of hearing impacts daily life, and get them to think more consciously about how audition plays a role in their lives. Students are then given an overview of the challenges of audition for humans, including the idea of trying to listen to what one person is saying in a room full of people talking, as well as the concept of sound discrimination and identifying the source and location of various sounds. This is then used to transition to the challenges of audition for machines, including appropriate hardware to correctly localize sound, and the software that allows computers to process sounds and take action based on what they hear.

3.2 Module 2: Signals (Pressure Waves)

In Module 2, middle school students become introduced to the concept of signals and waveforms, and begin to understand more about their properties. This module covers amplitude, frequency, and higher harmonics, and explains to students how each of these properties impacts signals, specifically in the context of music.

3.2.1 Pressure Waves

First, this module covers waves in nature to give students a visual understanding of what a signal looks like. This begins with examples such as a drop of water hitting a lake, and the concentric waves that ripple through the water as a result. This section also explores pressure waves that can be seen visually with particles. This gives students the fundamental understanding that audio signals have the shape of these physical waves, and this helps them understand the properties that are later introduced – amplitude, frequency, and higher harmonics.

3.2.2 Simple Waveforms

The curriculum then moves into signals more deeply, explaining the different properties of audio that humans hear: volume, pitch, and instrument type. These are tied back to the signal properties in the waveform itself, helping students understand what it is about the audio signal itself that makes it sound the way it does.

Amplitude and Volume

To begin, this section covers amplitude and how it contributes to the volume of a sound. Students are taught that the height of the wave from its midpoint is called its amplitude, and they also are introduced to the decibel scale of human hearing and how loud different common sounds are in decibels. They are then shown a video that plays various common sounds in order to give students a reference as to what the decibel values mean in practice. This is followed by a video explaining volume dynamics in sheet music, educating students about the dynamic markings: fortissimo (very loud),

forte (loud), mezzo-forte (slightly loud), mezzo-piano (slightly soft), piano (soft), and pianissimo (very soft). This prepares them for both the waveform visualizations given in Scratch, and the sheet music visualizations. These will be discussed in Chapter 5.

Frequency and Pitch

Students are then introduced to frequency and how it relates to the pitch of a sound. This begins by explaining to students that signals have a period, or amount of time it takes to go through one cycle. The inverse of this is frequency – how many cycles the signal goes through per second (Hertz). Students then learn that the faster a signal goes through cycles, the higher pitched the audio sounds. This is accompanied by waveform visuals in order to give students an understanding of what a signal looks like as it increases in frequency. Once this has been established, sheet music interpretation of this is also covered. Students are taught that the staff lines in music help users know the frequency of a note. The higher on the staff a note is, the higher pitched it is. This paves the way for both the waveform and sheet music visualizations that are used in Scratch.

Harmonics and Instrument

Finally, the last element of signals is higher harmonics. This section teaches students that the frequency that relates to the pitch humans hear is the lowest one present, or the fundamental frequency. However, the shape of the wave is what changes the sound of the instrument heard, even if it is playing the same pitch. This is accompanied by visuals and videos that show students how the waveform of different instruments look different, but since their frequency is the same, they sound like the same note. This section begins to introduce this through a frequency representation of a signal as well, in order to help students see the actual frequency content present in these new, complex signals. This is expanded upon in the next section, complex waveforms.

3.2.3 Complex Waveforms

This section covers the different frequencies present in an instrument's signals, and how the lowest frequency, or fundamental, contributes to pitch while the higher harmonics contribute to the distinct sounds different instruments make.

Fundamental Frequency

First, students learned that the frequency property discussed so far, relating to pitch, is referred to as the fundamental frequency. This is the lowest frequency present in a signal, and is the one that contributes to the note pitch that is heard. The Fourier transform visualization is then introduced, where students can see that they can view the frequencies of a signal on a graph of frequency vs intensity. This shows much more easily what frequencies are present, and it is no longer necessary to look at complex waveforms and try to estimate frequencies. Seeing this breakdown of frequency content allows students to have more in-depth conversations about what it means for a fundamental frequency to change, versus the higher harmonics. This transitions into the idea of harmonic frequencies.

Overtones

The concept of overtones, or higher harmonics, is introduced, and how they make each instrument sound the way it does. This is explained through the frequency representation of a signal, by showing students various plots of different instruments playing the same note, and emphasizing that although the fundamental frequency is the same, the intensity of the multiples of that frequency vary with instruments. The final visualization is introduced – spectrograms. These show frequencies over time through a colored graph, where the highest intensity of a frequency appears yellow, and the lowest intensities appear dark blue. These are used to show students how music can be visualized over time to see how notes change over time. This leads into a demonstration of different instruments playing the same note consecutively, and pointing out that each of them has the lowest frequency at the same place, but the

higher frequencies vary.

This module concludes the students' learning about signals and their properties, and from here they move on to understanding how these signals are processed by both humans and computers, going through the pipeline from sensation to perception.

3.3 Module 3: Sensation (Ear and Microphones)

In Module 3, students begin to learn more about the physical structures of ears and microphones, as well as all of the different hearing ranges that exist in different animal species. They begin to understand more about how machines process audio differently from humans, and how they follow similar patterns to humans as well. The module focuses on the actual sensing of audio in this chapter, making sure to emphasize the difference between this and perception, which is covered in the next module.

3.3.1 Input Devices

To begin this module, students begin to think about ears and microphones, and their similarities and differences. This lesson features an anatomical lesson about the human ear, as well as a lesson on the internal hardware of microphones on a high level. The goal is to compare and contrast the two, finding analogs in the structure of both which allows both ears and microphones to take in sound as input. This helps students understand how human ears physically sense sound, and how this compares to computer microphones.

3.3.2 Dynamic and Frequency Range

The lesson then moves into the idea of different ranges of hearing in different animals, as well as in different hardware. This begins with dynamic ranges, or ranges in volumes. For example, humans are only able to safely hear up to a certain volume, and also can only detect volumes above a certain decibel value. Students also learn about machine limitations, using the example of computational insect detectors and

how they can vary in picking up on different sounds based on their gain.

The lessons also introduce the idea of frequency ranges, or the pitches of sounds that different animals and devices can hear. For example, dogs are able to hear much higher frequencies than humans. Students discuss the hearing spectrum and where different animals' hearing ranges lie on the spectrum. They also discuss the variability in human hearing, namely that as people age, their ability to hear higher pitches diminishes. Infrasonic and ultrasonic hearing are quickly covered as well, citing examples from animals and machines.

3.3.3 Ear and Mic Analogs

This section then goes into the specific analogs between the human ear and a microphone. Specifically, students discuss the properties of both that allow for localization. In humans, it is explained that the outer ear contributes to the ability to localize, while machines rely on mic directionality and polar patterns to do the same. Students then discuss two types of transduction: the first which happens in the middle ear, or the mic diaphragms, respectively in humans and computers, while the second occurs in the cochlea or through magnets and coils in microphones.

3.3.4 Signals Revisited

Finally, analog and digital signals are compared, covering how the human ear detects analog signals while computers detect digital. This moves into discussing the human auditory nerve, which perceives continuous signals. Students then compare this to a machine's analog to digital conversion (ADC), learning that machines must take in audio by sampling it, in order to quantize and understand the sound that they hear.

3.4 Module 4: Perception (Brain and Computer)

In Module 4, students explore the way that both human brains and computers begin to actually understand the sounds they are hearing. This involves data representation,

as well as complex concepts that humans can do automatically such as sound localization, recognition, and understanding different auditory illusions. These processes are explained in humans, presenting the challenges in replicating them in computers.

3.4.1 Brain and Computer Overviews

The module begins with an overview of the processing power of both brains and computers – discussing the hardware behind computers, as well as various parts of the brain that contribute to the real-time processing of sound. In doing this, students build the foundation for talking more in depth about specific actions both can perform, and can move into data representation and perceptual abilities in both.

3.4.2 Data Representation

This section discusses data representation for both brains and computers. In brains, the engram is introduced, a unit of cognitive information that the brain uses to store memories. This is how students can think of brains storing data. This is compared to the computer's analog – storing audio. Students discuss the limitations of a computer and how its memory and processing power affects how it processes its data, and how it must store information digitally. This leads into sampling, or the rate at which samples are taken of audio so that it can be stored digitally, and how this can be a tradeoff of quality preservation and storage capacity. The lesson also discusses quantization, or having finite buckets that each value of a signal fits into so that each value can be stored in a finite number of bits. This is further discussed through resolution and bitdepth in audio storage.

3.4.3 Perceptual Abilities

The module then covers the perceptual abilities of both devices, and how humans can process sound through both their brains and computers. Students discuss the ability to recognize what a sound is, as well as the ability to localize sound. Finally, auditory illusions in humans are discussed, as well as adversarial examples in computers

performing machine learning.

3.5 Module 5: Human-AI Collaboration

The fifth and final module of the curriculum allows students to explore their new findings and use them to create a final Capstone project that involves music creation, processing, and accompaniment with a machine learning music library, Google Magenta [1].

3.5.1 Generative AI (Creative Collaboration)

Students are first given an introduction to generative AI, and the kinds of machine learning models used to create and accompany music. This begins with GANs, Generative Adversarial Networks. This model is able to create new data with the same properties as data it is given – hence, it is very useful for accompanying music that it has already seen, as it can generate music that sounds similar to what it has heard. Students are then introduced to RNNs, Recurrent Neural Networks, which can be used to generate new music on command. With these two new tools under their belt, students are then able to move forward and use the Scratch blocks we create with Magenta backend to work on their own machine audition projects.

3.5.2 Machine Audition Capstone

The end of the curriculum allows students to explore what they've learned and create their own AI music project. This includes using all of the Scratch blocks provided, including music creation and accompaniment, and allowing students to play with these to create and analyze their own music. This would also interface with some other activities, such as WaveGAN [7] and a Looper extension similar to MmTss [2] that would allow for more machine learning and interactive music creation concepts for students.

Chapter 4

Research and Design

This chapter discusses the design of the Scratch blocks, and how they were created around the machine audition curriculum in order to best complement the teachings with activities that were educational, creative, and suitable for middle school students. This project only developed Scratch activities for two of the five curriculum modules: Module 2, Signals, and Module 5, Human-AI Collaboration. The rest of the modules were much less computationally intensive, so it was determined that Scratch would not be needed for them. This chapter breaks down the activities that were designed, as well as how it was determined what Scratch blocks were necessary to accomplish the activities created.

4.1 Scratch Activity Design

This began by designing the activities that would be needed by students during the machine audition curriculum. This was broken down into the overarching topics students were expected to learn, and designed interactive, creative activities for each. The activities focused on signal properties and how they translated to music, with an additional activity for the machine audition capstone and music accompaniment concepts.

Live links and video walkthroughs of each activity listed below can be found in Appendix C.

4.1.1 Activity: Know Your Dynamics

This activity is designed to allow students to understand volume, and how it ties to the amplitude of a signal. It is explained to students that the dynamics of music convey information about volume. They are given two example sequences of music that play the same note – one where the volume increases with each subsequent note, and one where it decreases. These are then visualized in sheet music form, demonstrating the dynamic markings increasing with each note. This is meant to help students get acquainted with the sheet music visualization, and understand the dynamic markings a bit better. Students are also provided with a waveform of the sequence, demonstrating the amplitude increasing with each note as the volume increases, and vice versa. From here students move into an interactive activity. They divide into groups of 3 or 4, and at each person's turn, they play a note sequence that varies only in volume, and show the students just the final waveform or sheet music form of the sequence while hiding the Scratch blocks they used to create the sequence. From here, each student must attempt to recreate the sequence, and points are collected for each correct answer from the group. This activity aims to help students develop intuition for the connection between signal properties and how music sounds, and the hope is that students are quickly able to recognize that higher amplitudes mean higher volumes, and vice versa. In this activity, the goal is to look out for students getting relative volumes correct, more so than absolutes (i.e. a bigger amplitude followed by a smaller amplitude should result in a volume decrease, even if the student does not guess the exact volumes used.)

A screenshot of the final Scratch activity can be seen in Appendix B-1. A live link can be found here. A video demonstration of this activity in use can be found here.

4.1.2 Activity: Making Melodies

This activity moves into pitch and frequency of music. The instructors demonstrate a major and a minor scale, and visualize both for the students. Here, they can see that the sheet music visualizes the notes higher up on the staff as the scale progresses, and

the waveform visualization showcases higher frequency waves as the notes go up in pitch. From here, students are divided into groups, and allowed to create their own melodies. They create first melodies in the key C major, and visualize the sheet music and waveform for this to show how the visualizations change as the frequencies change. They also do this for a second melody in C minor, and analyze the visualizations that come out of it. Students are able to discuss the reasons the visualizations change as they modify their tunes, solidifying their understanding of pitches and how they relate to frequencies in music. Students do not do the same group guessing game as seen previously, since it is much more difficult to see the differences in frequencies, and the waveform legend also shows students the note names that were used for ease of understanding the plots.

A screenshot of the final Scratch activity can be seen in Appendix B-3. A live link can be found here. A video demonstration of this activity in use can be found here.

4.1.3 Activity: Find Your Voice

This activity also focuses on pitch and frequency. In this activity, instructors help students find their vocal range by singing along to a premade sequence of notes that simply steps through the scale. These notes are also visualized in intervals so students can see them in frequency and in sheet music visualization. Students hit play on the sequence, and sing along to an increasing set of notes until they cannot hit the note anymore. They repeat the process for decreasing frequencies until they hit their lowest note. Students can then come up with their vocal range, and what voice classification they have. Although this activity does not focus as much on signal properties, it does help students gain intuition for these frequencies as they attempt to imitate the sounds themselves.

A screenshot of the final Scratch activity can be seen in Appendix B-2. A live link can be found here. A video demonstration of this activity in use can be found here.

4.1.4 Activity: What's Your Timbre?

This activity focuses on the higher harmonics of different instruments that lead them to sound different despite playing the same notes. Instructors begin by playing the same note on a few different instruments, and allowing students to hear the difference between them even though the notes are the same. The instructors then visualize the waveforms for both instruments, and show students that, although they can see the notes are the same frequency, the shape of each waveform looks noticeably different. The students are then introduced to the frequency visualizations - Fourier transform magnitudes and spectrograms. Here, they are able to decompose the waveforms into their frequency components, and can see that the lowest frequencies match, but the higher harmonics look very different. Students are able to understand that the fundamental frequency is responsible for pitch, while the higher harmonics are responsible for timbre. Students then experiment with this themselves, selecting different instruments and notes to visualize so they can see how the waveforms differ with different instruments. Instructors make sure to emphasize here that the fundamental frequency will not change if they simply change the instrument type, but it will change if they select a different note to play. This is meant to help students learn to differentiate fundamental frequencies from higher harmonics visually, as well as tie it back to how the music sounds.

A screenshot of the final Scratch activity can be seen in Appendix B-4. A live link can be found here. A video demonstration of this activity in use can be found here.

4.1.5 Activity: More Bars, More To See

This activity is the second to focus on the timbre of different instruments. This one allows students to begin making longer pieces of music, including verses and choruses. Instructors start by showing students a 16 bar verse, and playing it for them. They then display sheet music, as well as a spectrogram to show sheet music over time. Then students are able to create their own 16 bar verses. They are asked to note what timbres were used, and visualize their songs in spectrogram and sheet music form.

Here, students are able to see that they can only extract pitch information from sheet music, but spectrograms are also able to display higher harmonic information that lets them know when the instrument is changing. From here, students are able to better understand how harmonics change the sounds they hear from different instruments more intuitively.

A screenshot of the final Scratch activity can be seen in Appendix B-5. A live link can be found here. A video demonstration of this activity in use can be found here.

4.1.6 Activity: Machine Audition Capstone

Finally, students are able to incorporate what they have learned about signals with some human-AI collaboration elements. To do this, they are given the ability to create their own music, and select instruments and volumes for each note they add. The goal is to make this an intuitive, streamlined process so they can seamlessly create their music. From here, they are able to visualize their music in waveform, FFT, spectrogram, or sheet music format. To include human-AI collaboration, students are given access to Google Magenta [1], a library that is capable of machine learning for music generation. Students should be able to generate and accompany their own music, and be able to analyze it in order to further solidify their knowledge from the machine audition curriculum.

4.2 Scratch Block Preliminary Design

After creating the activities, it was possible to come up with a list of Scratch blocks needed as well as their functionality. These blocks were divided into 4 different groups based on functionality: analysis, visualization, creation, and accompaniment.

A table of the initial Scratch block design and functionality can be seen in Appendix A.1.

4.2.1 Music Analysis

First, students needed to able to analyze music in order to extract its properties, as well as compare various mystery files to determine their properties relative to each other. It was determined that students would need blocks that could return the volume, pitch, and instrument type of a given sound file, as well as blocks that were able to return which of two mystery files was louder or higher pitched. Students also needed blocks that would report the instrument each of the files was using.

4.2.2 Music Visualization

Students also needed to be able to see the proper visualizations in Scratch real-time. This meant including a waveform, an FFT magnitude, a spectrogram, and a sheet music visualization. These visualizations also needed to have some sort of labeling so students could pick out the important properties – amplitude, frequency, and higher harmonics. They needed to be designed such that all the visualizations remained up-to-date as the user created their music, and that they would be consistent with each other every time they were called.

4.2.3 Music Creation

The bulk of the interactive element of these activities came from allowing students to create their own music. Students needed to be able to select a volume and an instrument type at any given point of creating music, such that the music they created after that point had the selected properties. It was also necessary for students to be able to quickly see what instrument and volume they had last selected, so they could tell if they needed to use a new Scratch block to update any of these properties. It was also important to make the process of adding notes to music as seamless and intuitive as possible. This meant giving students some way of hearing the notes they were selecting before they were definitively chosen for their song. Thus, it was necessary to include a block that would allow students to choose their notes from a virtual note player that would allow them to hear the note before selecting it, and give them

a duration field that allowed them to set an amount of beats to play the note for. Finally, students needed to easily be able to erase all of their music and start again.

4.2.4 Music Accompaniment

Finally, students needed the ability to perform music accompaniment with the help of machine learning. This simply meant giving them access to Google Magenta in order to both generate brand new music, as well as accompany already created music. They also needed the option to select how fast-paced they wanted the resulting music to be, as well as what duration of music they wanted. the goal was to give students some amount of information about how these blocks worked, and introduce them to the general concept of machine learning, without overwhelming them with machine learning content in a primarily signal processing curriculum. Thus, the blocks needed to abstract away the machine learning backend, and simply give students the ability to select a block that would generate or accompany music on command, leaving the harder machine learning concepts to Scratch. This meant that the extension needed to integrate Magenta fully into this section of the blocks, such that users would not have control over the actual machine learning backend, and could simply use the blocks to generate music at will.

The implementation of each of these blocks will be covered in Chapter 5.

4.3 Backend Library Research

After designing the Scratch block functionality needed, more research was conducted on existing backend JavaScript libraries that could be used in Scratch in order to minimize redundant work. A few crucial libraries were found that were used in the Scratch backend and allowed higher-quality, quicker development of the blocks. It was also necessary to get familiar with Scratch's current functionality, and determining how to use the existing functions in the machine audition extension's backend. Namely, it was decided to utilize the Scratch Pen extension as the main backend for visualization, such that it could be used to write wrapper functions to provide plots

and sheet music given a music data structure. The Scratch Music extension was used as inspiration for the Music Creation blocks, to recreate a similar backend that also added new functionality and different data storage such that it would both accomplish the machine audition education goals, as well as offer a more optimal way of representing data for the functionality needed.

Various DSP libraries were also researched for the signal processing portion of the Scratch blocks, but it was determined that it would be better to do this manually. This was due to the fact that DSP libraries are meant to analyze real-world signals, and will introduce various artifacts that are a result of sampling rates, real-world noise, finite signal discontinuities, etc. Because it was important not to confuse students with these concepts, the signal processing functions were designed from scratch to perform an "ideal" signal analysis, such that each frequency was represented as a perfect delta in frequency domain rather than having any artifacts of real-world signals. This would allow students to focus more on the theory behind signal processing and understanding music, as opposed to learning about all the different ways signals can change in the real world.

4.4 Scratch Asset Design

Because this curriculum was targeted at middle schoolers, it was important to ensure that the visuals and storyline behind the curriculum was engaging for the target age group. As such, the curriculum was designed to follow a character's story, and build out a backstory that would guide the students along the curriculum, providing context and engaging stories as well as attention-grabbing badges as they completed different modules and activities. This was made possible by a freelance designer who also designed the assets for the Sight of AI Curriculum. We were able to come up with a fitting character and robot companion to build the curriculum around.

The curriculum ended up with a character named Lucas and a robot companion named Audii, who captured the concept of machine audition through his design. We see a visual of Lucas and Audii in Figure 4-1.

Figure 4-1: Lucas and Audii.

These characters are used throughout the curriculum, with the backstory that Audii was discovered by Lucas in an old abandoned warehouse. As students progress through the curriculum, Lucas and Audii gain new knowledge and skills, as they each understand their own senses of hearing, human and machine.

Chapter 5

Implementation

This chapter covers the implementation of the Scratch blocks for the machine audition curriculum. The blocks were added as an extension to Scratch, written in JavaScript. The blocks are divided into four categories that share a backend to maintain real-time concurrency. The blocks fit into either music analysis, music creation, music visualization, or music accompaniment. First, this chapter will explain the structure of the code as a whole and how these sections interacted with each other before describing the backend implementation and system architecture of each individual section.

Two different types of Scratch blocks were implemented in this extension – command blocks and reporter blocks. Command blocks are primarily utilized in cases where the user is modifying or creating new functionality, while reporter blocks provide users with information that has already been stored, and do not directly alter the code.

5.1 System Architecture

The Scratch blocks created all share the same backend and data structures, in order to ensure that the blocks maintain the same information in real-time. It is important for students to be able to create music that allows them to modify instruments and volume before adding a note with the properties they selected. They also needed to

be able to view the properties they had selected at any given time, so these variables needed to be stored separately from the music the user was creating. Finally, it needed to be ensured that any music that was created with machine learning could also be added to the music the user had added and played in the same way. Both the music creation and accompaniment results then needed to be shared with the visualization backend, so students could analyze their own creations. This resulted in two separate system architectures – one combines music creation, visualization, and accompaniment, while music analysis was a separate system since it did not have any information overlap. This chapter will break down both of these systems in the following sections – first Music Analysis in its free-standing form, and then how the other three modules were structured and how they worked together.

5.2 Music Analysis

In this set of blocks, students need to learn intuitively how to hear the difference in signal properties. The goal is to give them the opportunity to compare two mystery sound files, each playing one note each, and ask Scratch about the properties of each sound file. For example, students should be able to ask which of the two files is louder, which is higher in pitch, and what instrument is being played in either. To do this, the backend needed to be able to play and analyze both files that were requested by the user, as well as update reporter variables as to which was louder, higher, and what instrument each was, so the user could simply request an answer to one of these questions. These blocks did not need to have the same backend as the other categories, as they did not overlap in information. The students needed to be given simple music examples, in order to help them begin to isolate the different signal properties and their impact on music.

5.2.1 Music Analysis Backend

The Music Analysis blocks were intended to perform controlled comparisons between single note sequences to allow students to explore how signals properties change the

music they produce. The system diagram for these blocks can be seen in Figure 5-1.

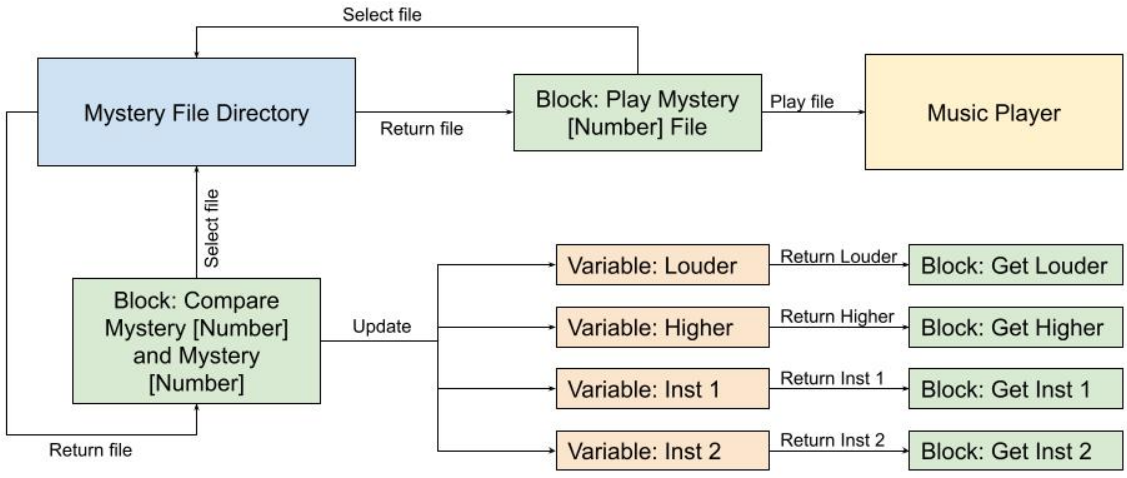


Figure 5-1: Music Analysis System Diagram.

In these blocks, a backend directory of 4 different single note files at different instruments, volumes, and pitches for students to select from. Upon selecting a file, students could implement a block that played the file by way of interfacing with a player that had access to each of the backend files. Students could also use the comparison block and select two mystery files from the dropdown to compare, at which point they could toggle reporters for the instruments played, as well as which was louder or higher. The way these properties are determined can be seen in the system diagram above – because they were premade files, the code could store premade answers for each of these blocks. Every time a student ran a new comparison block, the backend would update each of the reporter variables, so that they could be updated in the user's view. Figure 5-2 displays the Scratch blocks, as well as the

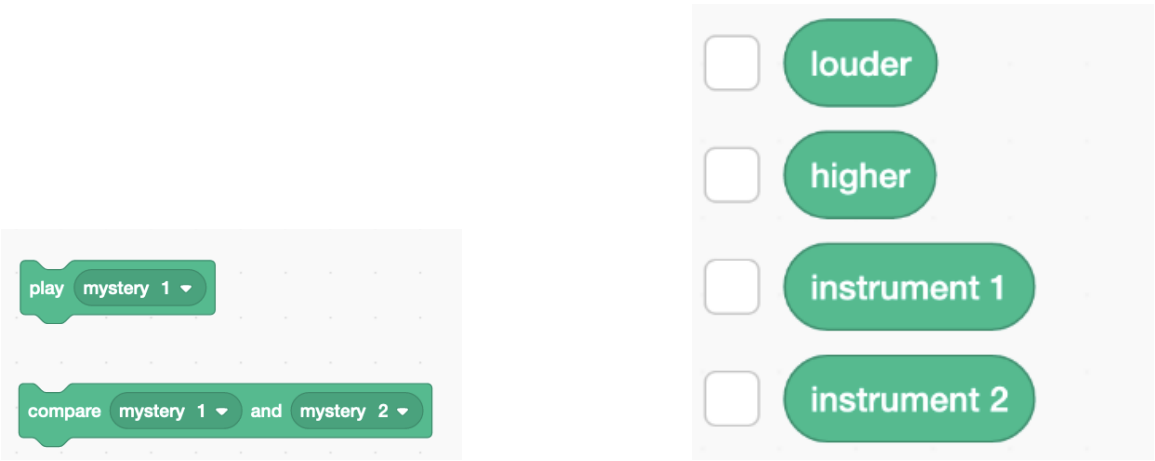


Figure 5-2: Music Analysis Blocks and Reporters.

reporters that are toggled on and the resulting answers that appear in Scratch.

See Appendix A.4 for a full table of the Music Analysis blocks.

5.3 Music Creation, Accompaniment, and Visualization

The music creation, accompaniment, and visualization blocks all needed to share information such that the blocks would all have updated music as the user changed and added to their music. In the next few sections, each of these systems are explained conceptually, before going into the backend of each. It will then be explained how these systems communicated information to each other as well, and how system updates were optimized to be as infrequent as possible while still maintaining consistent information across all subsystems.

5.4 Music Creation

In this set of blocks, students needed to be able to create music by adding notes in a sequence, and change the volume and instrument that is playing each note. To do this, the extension needed a way to store the music the user has added so far, as well as keep track of what the volume and instrument settings were when each note was

selected. This way, it would be ensured that each selected note would be played at the correct volume and by the correct instrument. Students also needed to be provided with a way to clear their music and start from scratch. Finally, students had to be able to see what the current volume and instrument choices were so they could know whether they needed to make a change with a set volume or set instrument block. This would allow for better usability, and lower redundancy in users' code.

5.4.1 Music Creation Backend

Music Creation is the foundation for both music accompaniment and visualization. Figure 5-3 shows a system diagram of the music creation subsystem.

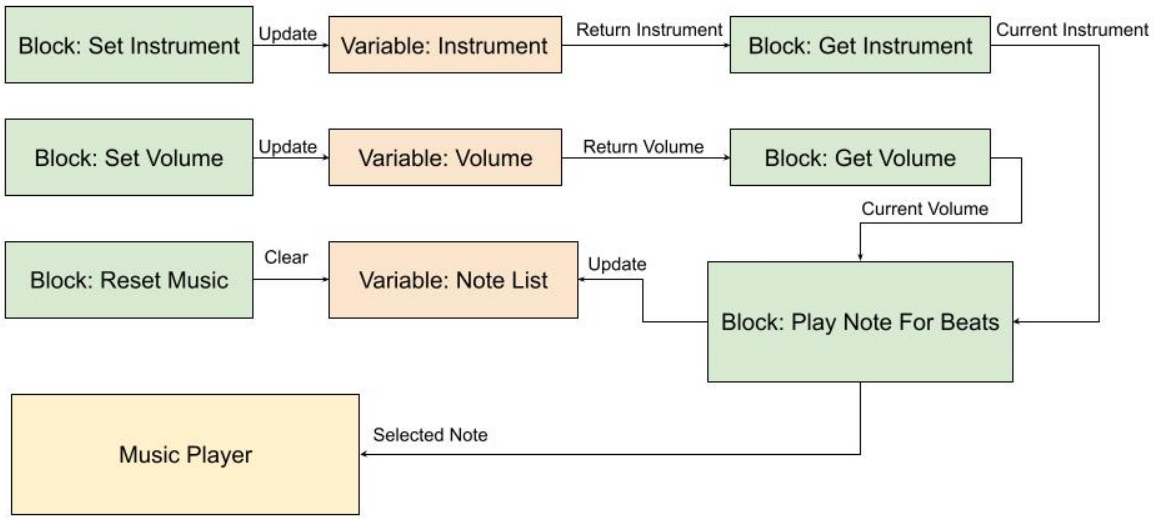


Figure 5-3: Music Creation System Diagram.

A few important variables are kept track of here: the note list is used to build up the music the user has created so far, and the volume and instrument variables

are used to create notes with the correct properties as the user adds them. Users can update these two variables with dropdown menus, and set volumes in terms of dynamic markings (fortissimo, pianissimo, etc.) as well as select from a chosen list of instrument options. This will then update the volume and instrument to use for every subsequent note the user plays. The user can then use the play note block to select new notes – this will display a virtual piano dropdown that the user can play notes on before making their selection. From here, the Scratch block will simply append to the note list a tuple that consists of the note frequency the user selected, the note duration that they gave in beats, the current instrument, and the current volume. It will also play the note that has been selected by the user in the correct volume and instrument when they click the Scratch block. When the user presses the reset music block, the note list is emptied and ready to build new music. When the user presses play on the blocks, the note list will be used to synthesize the music. This is done by using the Scratch Music backend in order to play the correct instruments at the correct durations, pitches, and volume.

Two reporter blocks are also included – current volume and current instrument. These allow the user to see what the last selected instrument and volume were, such that they can easily check what volume and instrument any new notes will be played as, so they can then change them if needed. This supports larger scale music creation and allows users to work more efficiently.

In Figure 5-4 a visual is shown of each of these Music Creation Scratch blocks, as well as the reporters that are included for users to keep track of instrument and volume choices.

A list of music creation Scratch blocks can be seen in Appendix A.3.

5.5 Music Accompaniment

This set of blocks was simply meant to interface with Google Magenta. The goal was to give students the option to randomly generate entirely new music, or auto-complete music they had already created, or accompany the music they had created.

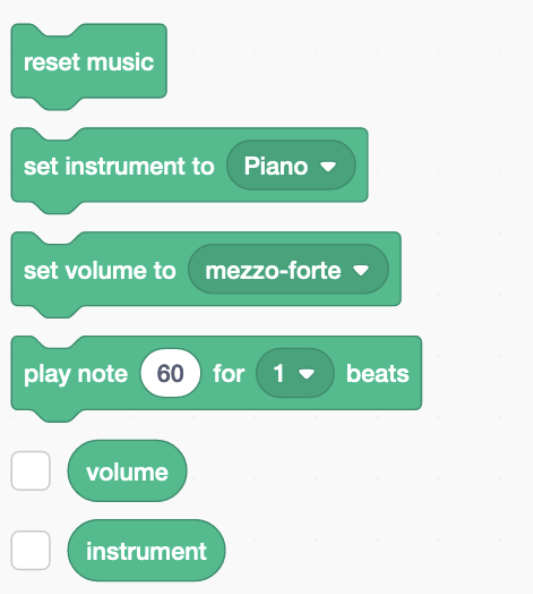


Figure 5-4: Music Creation Blocks and Reporters.

It was also important for students to be able to see visualizations for this created music, and integrate it with their manually created music if possible. Thus, the music accompaniment blocks also needed to see the same data structures as music creation so they could add to the same music, and it also needed to be handed off to music visualization so students could see what their music looked like with the visualization blocks.

5.5.1 Music Accompaniment Backend

The music accompaniment blocks give users the ability to generate, complete, or accompany music. Users are given the ability to modify the length of music that is created, as well as the temperature, or how quickly the notes change in the music. The system diagram for this system can be seen in Figure 5-5.

When Magenta is called, it is able to return and play the music that it created from the inputs that the user provided. If the user is using the generation block, the created music note list is rest to turn into the Magenta output. If the user chooses to complete their music, the Magenta output simply gets added to the already existing note list. This Magenta music can then be played once the code has finished running,

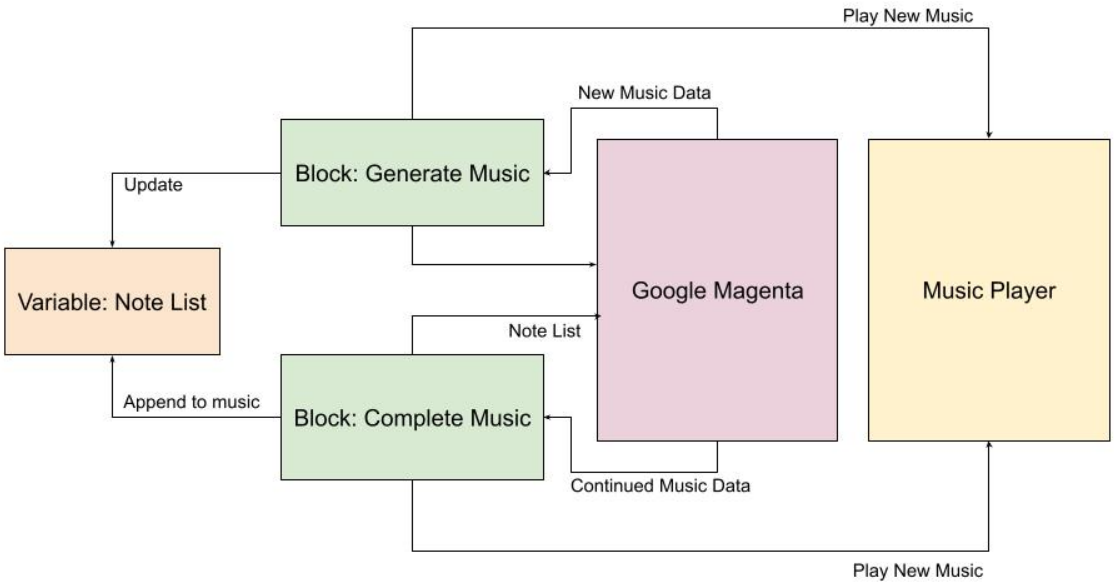


Figure 5-5: Music Accompaniment System Diagram.

and users can listen to their new generated music the same way they do in the music creation subsystem.

Figure 5-6 shows a visual of each of these Music Accompaniment Scratch blocks.

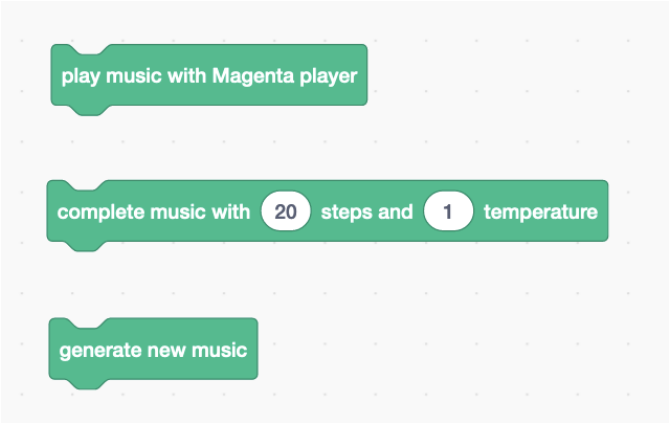


Figure 5-6: Music Accompaniment Blocks.

An example of the Magenta music generation block can be seen in Figures 5-7 and 5-8.

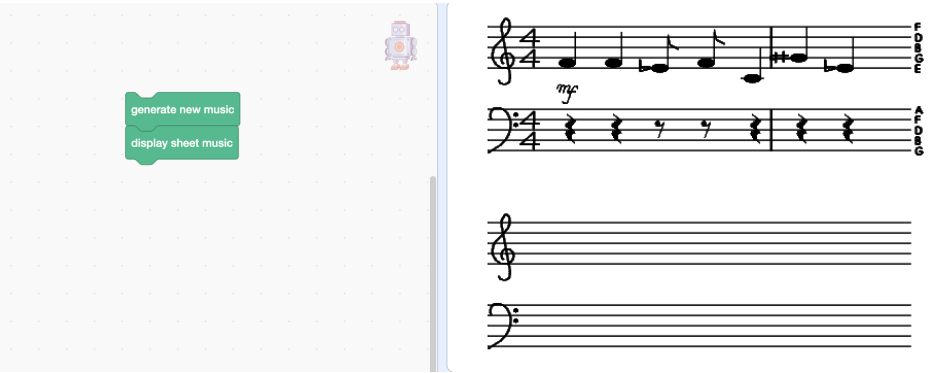


Figure 5-7: Music Generation Sheet Music Visualization.

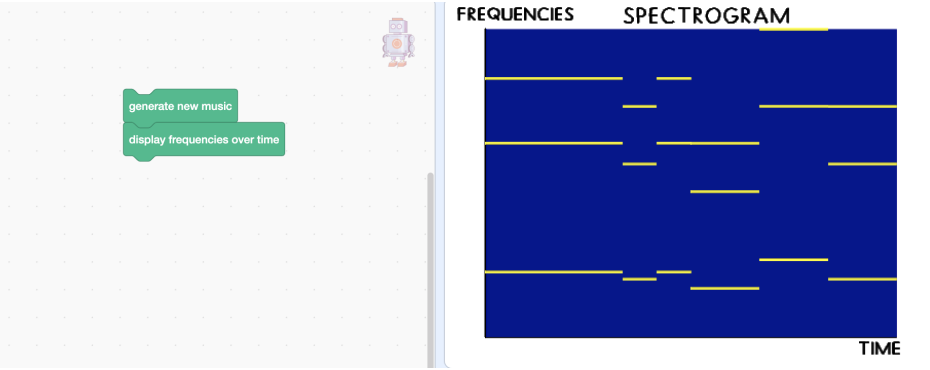


Figure 5-8: Music Generation Spectrogram Visualization.

An example of the Magenta music completion block can be seen in Figures 5-9 and 5-10.

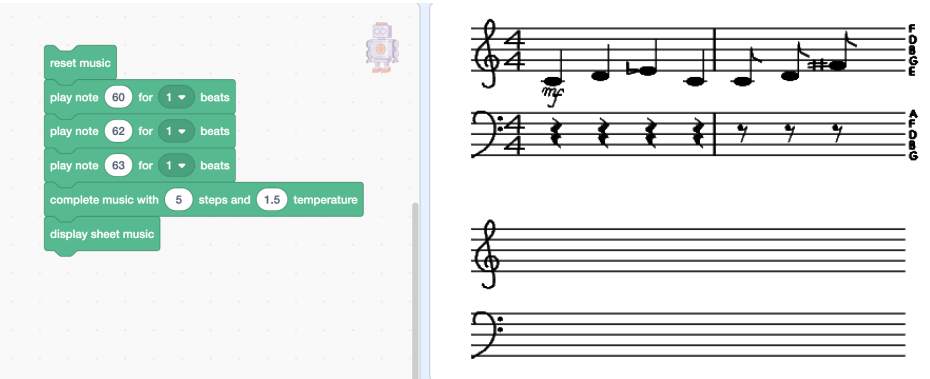


Figure 5-9: Music Completion Sheet Music Visualization.

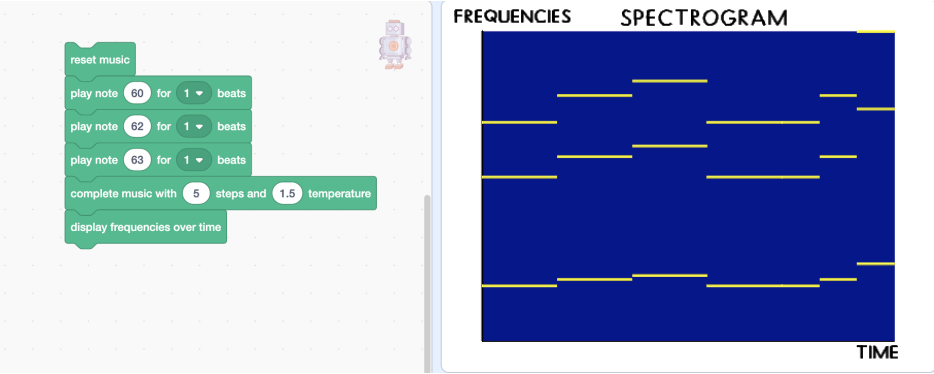


Figure 5-10: Music Completion Spectrogram Visualization.

A list of music accompaniment Scratch blocks can be seen in Appendix A.5.

5.6 Music Visualization

In order to support all of the ways to view music, it was important to allow students to see a waveform, Fourier transform, spectrogram, and sheet music visualization for any music they had created either manually or through machine learning with the Google Magenta backend. To do this, the visualization blocks needed access to the music creation data structures in real-time, such that they could visualize the music the user had created correctly. Each visualization also needed to restructure the note sequence in a different way based on how it used the information. For example, sheet music visualizations needed to know the note conversion from a piano key to a note name, and durations needed to be converted to a note type (quarter, eighth, whole, etc.). Spectrograms, on the other hand, required a Fourier transform given the instrument that was playing the note, and this then needed to be converted into a sequence of frequencies over time to be plotted. Music visualization also needed to incorporate the music accompaniment results as well, if the student had generated their music or added to it with Google Magenta. Each of the four visualizations is explained in depth below.

5.6.1 Music Visualization Backend

With the music creation and accompaniment done, the music lists now needed to be represented in four different ways: waveform, Fourier transform, spectrogram, and sheet music. The system diagram for the music visualization backend can be seen in Figure 5-11.

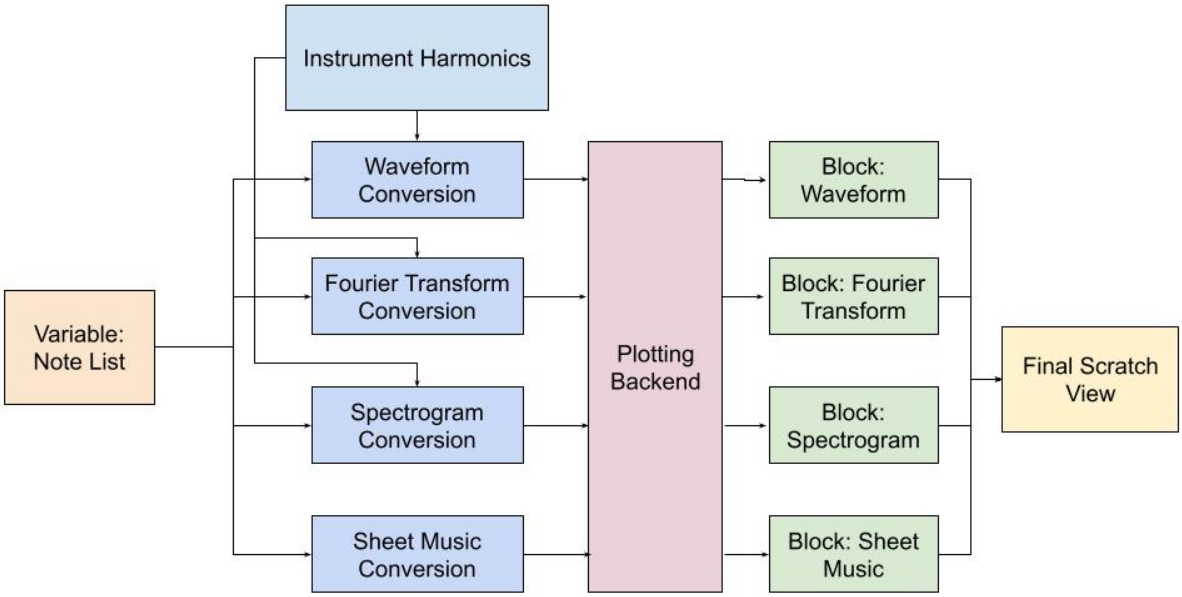


Figure 5-11: Music Visualization System Diagram.

There is a fair amount of overlap between the visualization types that Scratch provides. These are broken down separately, noting the overlaps throughout.

Waveform

The waveform visualization plotted a time-domain signal that represented the music the user had selected. To do this, the note list first needs to be converted into a series of frequencies, number of samples, and an amplitude. Each piano key is

converted to its frequency value in Hz, and the duration in seconds is converted to a number of samples such that the resulting graph will look continuous. To do this, every duration was multiplied by 10000, essentially aiming to have 10000 samples per second. Finally, the amplitude was simply a scale factor where 1 was fortissimo, and 0.1 was pianissimo, and the values in between followed a log scale to reflect the human hearing range. Once these values had been found, for each note, the code looped through the number of samples, and computed the value that would correspond to its fundamental frequency, according to Equation 5.1.

$$x[n] = \sin(\Omega n) \tag{5.1}$$

Here Ω is defined to be the corresponding frequency found, and n is the sample being plotted. This creates a simplified wave that is still visible by the user in Scratch once it is plotted, and makes it easy for users to determine how long the period is. They are still able to see where frequencies go higher vs. lower, and when amplitudes increase or decrease. Finally, the harmonic frequencies need to be added. The code has stored a dictionary for each instruments' higher harmonic content, where the harmonic multiples have been mapped to an intensity. This is then used to build up on the original fundamental frequency with an equation that looks closer to Equation 5.2, and mimics a simplified Fourier synthesis equation.

$$x[n] = \sum_{k} h[k] \sin(\Omega k n) \tag{5.2}$$

Here h[k] is defined as the harmonic intensity, and k is the harmonic multiple itself. This builds up a waveform that has the correct fundamental frequency, and showcases each instruments' unique wave shape.

Once the code has all of these samples, the result is a list of sample indices and sample values. Plotting can then be implemented to put each of these on a graph in Scratch. Care is also taken to change the line color every time a switch is made to a new frequency in order to ensure students can make out the differences. If a frequency is repeated, it is plotted in the same color as all other portions of the wave that were

the same frequency. The frequencies are labeled in a legend in the upper right corner, so students can know what note made each frequency. Finally, the graph is labeled as a waveform visualization, and the axes are labeled as sample versus amplitude.

A visual of the Scratch waveform can be seen in Figure 5-12.

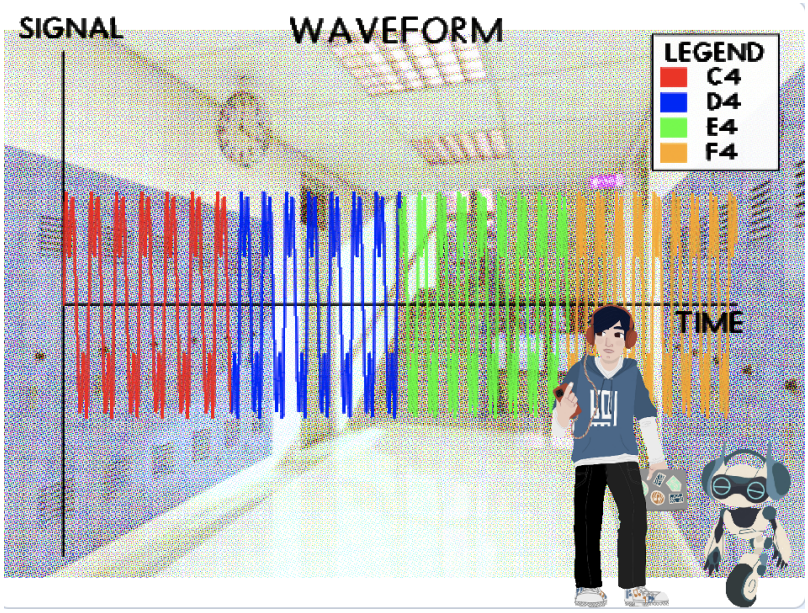


Figure 5-12: Waveform Visualization.

The waveform has a few constraints – first, it can only visualize 4-5 notes at a time. This is due to the fact that adding more notes would make the graph cover too much time, and it would become increasingly difficult to make it visually look like it had different frequencies. It is proposed that if a user needs to see more music than just 4-5 notes, they could switch to a more space-efficient representation, like sheet music or spectrograms, both of which do not need to sample a large amount of the signal to show any useful information.

Fourier Transform

To do a Fourier transform, a modified, simpler version was implemented in order to reduce any artifacts that would be seen from a real Fourier transform. Firstly, the frequency values themselves were immediately converted to a value in Hz that would represent the fundamental frequency. From here,the backend data structure

that stores each instruments' harmonic values and intensities was used to generate the harmonic information for each note, scaled by the fundamental frequency being played. The code was then able to plot an XY axis where the X represented the frequency, and the Y represented the intensity. The same plotting backend as the waveform was then used in order to plot a discrete stem plot of points that represented these frequency/intensity pairs. If there were multiple notes that had the same frequency, the intensities for those frequencies added, since this is what would happen in an ideal scenario where windowing and finite-length signals would not contribute to artifacts and potential aliasing.

A visual of the Scratch Fourier transform can be seen in Figure 5-13.

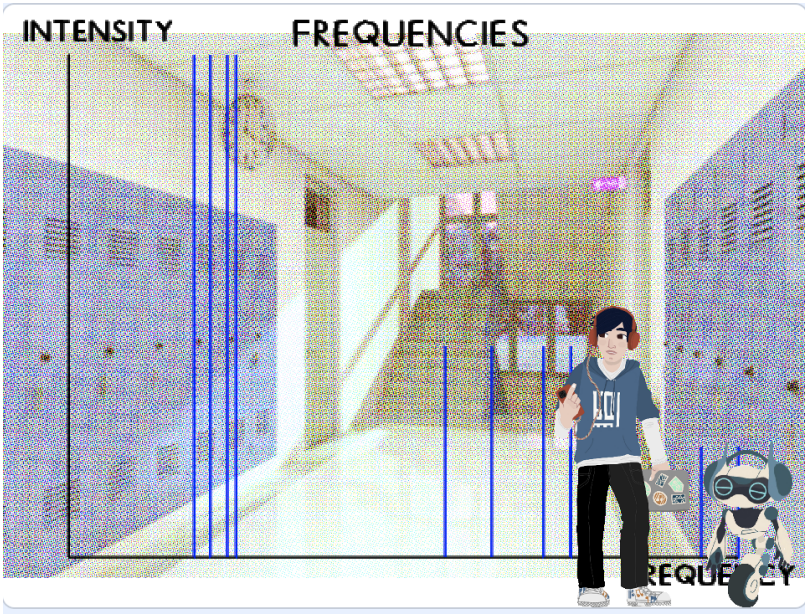


Figure 5-13: Fourier Transform Visualization.

The Fourier transform is limited in that it does not currently have color coding. This makes it difficult to tell what frequency is coming from a fundamental versus a harmonic frequency. This could be changed such that fundamentals would have one color while harmonics had another, or fundamentals were color coded to match their harmonics. It is also lacking in a legend that could help students better understand what frequencies they were seeing in this graph.

Spectrogram

The spectrogram followed closely from the Fourier transform, however, it also needed to keep track of timestep information. For each note, the code is able to extract the fundamental and harmonic frequencies by the same method as seen in the Fourier Transform visualization. However, it also keeps track of the start and end time in beats of each of these notes. From here, the spectrogram is plotted on an XY axis, where the X axis represents time, the Y axis represents frequency, and the intensity is shown as a color value where dark blue is 0 intensity, and yellow is highest intensity. The spectrogram is then plotted such that the entire background is painted dark blue, and each frequency is represented as a horizontal yellow line at the corresponding Y value, stretching from the start to end time value. This two-toned graph does not represent a spectrogram for a more complicated, real-world signal, as it would end up with more smearing and variety of color in a real-world signal. However, this simplification allows students to gather the information they need without being overwhelmed by the artifacts caused in the real world.

A visual of the Scratch spectrogram can be seen in Figure 5-14.

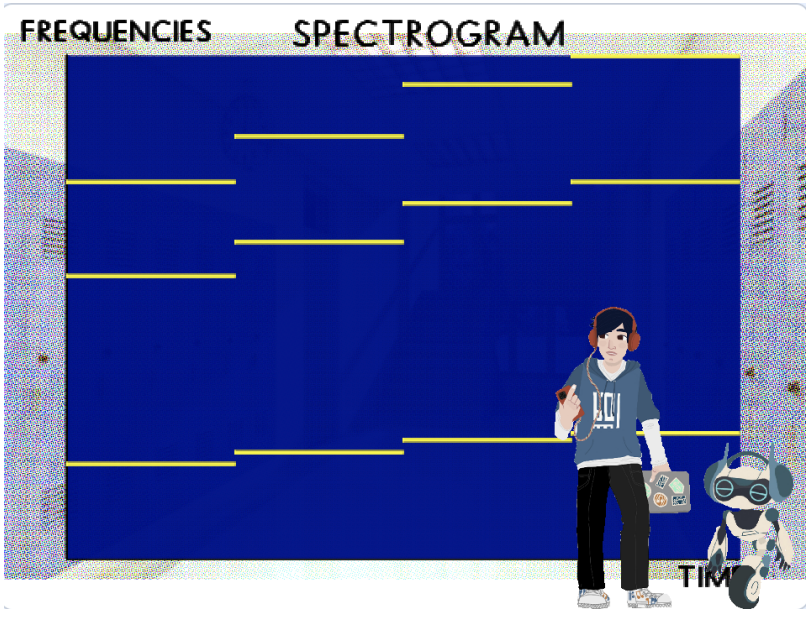


Figure 5-14: Spectrogram Visualization.

The spectrogram was less limited than other visualizations, since it maintained

the time information, did not require a lot of space for each note, and was also able to preserve higher harmonic information so students could see when instruments changed, versus when the note itself changed. It could use more labeling to explain to students what each frequency represents, but otherwise preserves accurate and easy to read information.

Sheet Music

Finally, sheet music was implemented for this project. To create this, the note list first had to be converted once more. Here, the piano notes were mapped to a corresponding place on the grand staff, where it was noted whether the note was in treble or bass clef, and then what spacing it would fit on (0 being just below the bottom line, 1 being intersected by the first staff line, etc.) The code had to map which note type each duration would get (whole, half, quarter, etc.) based on the number of beats the user had selected. Once this was done, the grand staff lines were drawn by simply creating the set of 2 staffs, 5 lines each, to represent the grand staff. The code also used data points to construct a treble and a bass clef, as well as a time signature that was made common time as default. From here, each note that had been played in the music was plotted on the sheet music in its corresponding note type. This implementation also dynamically kept track of how many beats had been used, and when it hit 4 beats, drew a measure bar. If a note was too long to fit into the measure, ties were employed to split up the note length into two, such that the first half would be the end of one measure and the second half would be the beginning of the next measure. The music also included dynamic markings for each note, writing out the music term for the volume that the user had chosen every time it switched from a previous setting. The code also ensured appropriate rests in the opposing clef rather than leaving the other clef empty.

A visual of the Scratch sheet music can be seen in Figure 5-15.

The sheet music could have a few more changes made to it to improve – first, there can be more dynamic rearranging of rests such that the rest durations will add up if there are multiple right after each other. Furthermore, more support should be

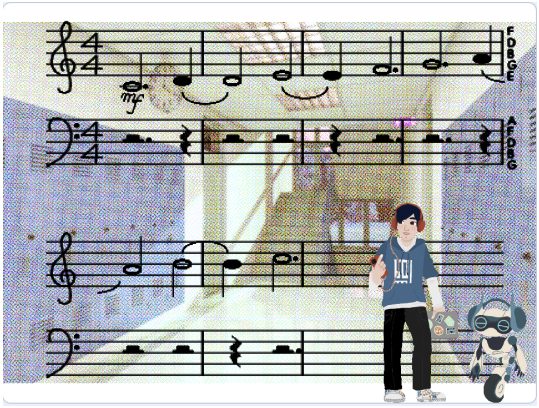


Figure 5-15: Sheet Music Visualization.

added for more complicated note ties, as currently this can only support ties up to eighth note duration. Support could also be expanded for different time signatures. Otherwise, the sheet music is performing well as is.

The visualization blocks can be seen in Figure 5-16.

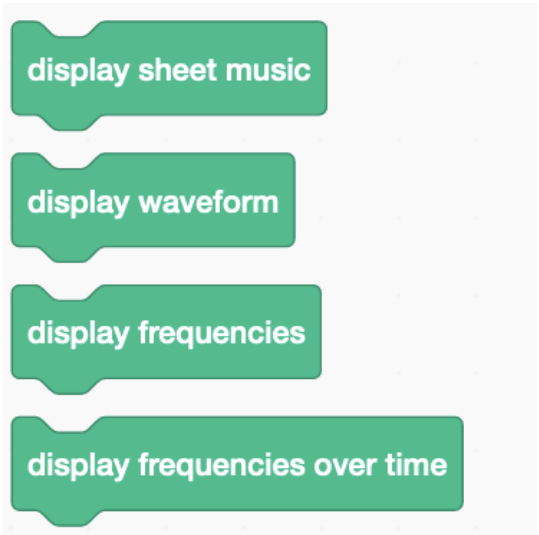


Figure 5-16: Music Visualization Blocks.

A list of all of the music visualization Scratch blocks and their functions can be seen in Appendix A.2.

5.6.2 Plotting and Text Implementation

Two main contributions to Scratch that this project provided were plotting, as well as text rendering. Here the backend for both of these systems is broken down, as well as how they were integrated into the Machine Audition extension.

Plotting Implementation

Implementing plotting was one of the biggest challenges of this project. Scratch prior to this project did not have plotting capabilities, so an entire framework for plotting was written within Scratch for this project. A system diagram for plotting can be seen in Figure 5-17.

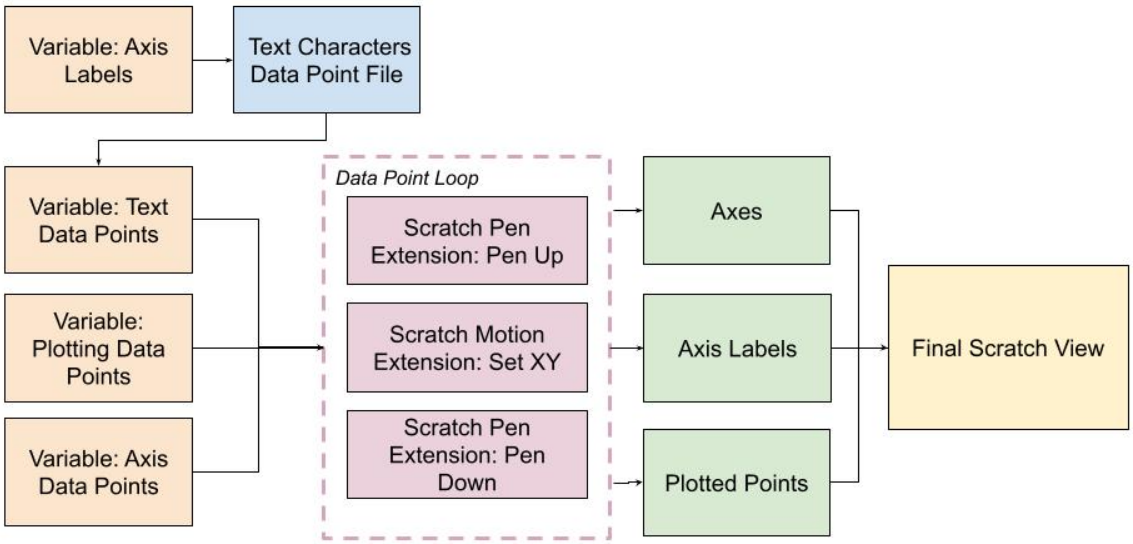


Figure 5-17: Plotting Backend System Diagram.

The plotting backend primarily used the Scratch Pen extension to do this. The code had tie a pen to an invisible sprite, and use the backend pen controls as well as

the sprite movements to create functions that would draw things like axes, legends, continuous and stem plots, and sheet music. The pen extension was used to control the color of the lines, as well as whether the pen was up or down (writing or not). It then had to use the sprite movement controls in the backend such that the sprite would follow the path of the pen, creating the drawings needed. In the future, plotting would be separated from Sprite movement, but currently Scratch does not support this.

In order to implement plotting, an input list of (x, y) coordinates was used, which had been post-processed from a note list data structure. This ensured that the plotting function itself was normalized to work for any set of data. From here, the values were normalized on the x and y axes such that the values would fill up the entire space of the graph. Once had this new list of data points was created, the code needed to follow a loop over the data points to make sure to raise the pen, set a new XY, and lower the pen, before drawing a connecting line to a new point by setting a new XY again. This loop then created the axes, the axis labels, and the plotted points that were displayed by Scratch.

A legend was also implemented – this was useful for the waveform visualization in particular. A user could specify the label they wanted, as well as the color they wanted for each label, and Scratch would create a white box with black outline labeled 'Legend' in the top right corner. From here, it would use the pen extension to draw a square of each color the user had requested, and use the text rendering backend to write the label next to each. These colors were then used in waveform visualization to ensure that the plot changed colors when the frequency changed, as this was how it was made easier for users to tell the different frequencies apart. This could also be added to the Fourier Transform visualization such that users could keep track of fundamental vs. harmonic frequencies better.

This was used for the waveform, Fourier transform, and spectrogram visualizations. Separate functions were written for the actual content of each plot, as the process varied for each. A separate initializing function was created for sheet music, as it needed to include staff lines, as well as the clefs, notes, and dynamic mark-

ings, which were all implemented through text rendering. Overall, it was possible to combine the axis creation for Fourier Transform and spectrogram, and the plotting functionality for waveform, spectrogram, and fourier transform depending on the different types of plots wanted, but the different visualizations branched off from here.

Text Rendering Implementation

Text rendering was the next important contribution to Scratch. This method was used not only for text and labeling visualizations, but also for creating all of the symbols in the sheet music (different notes, clefs, dynamic markings, rests, etc.). A system diagram of the text rendering backend can be seen in Figure 5-18.

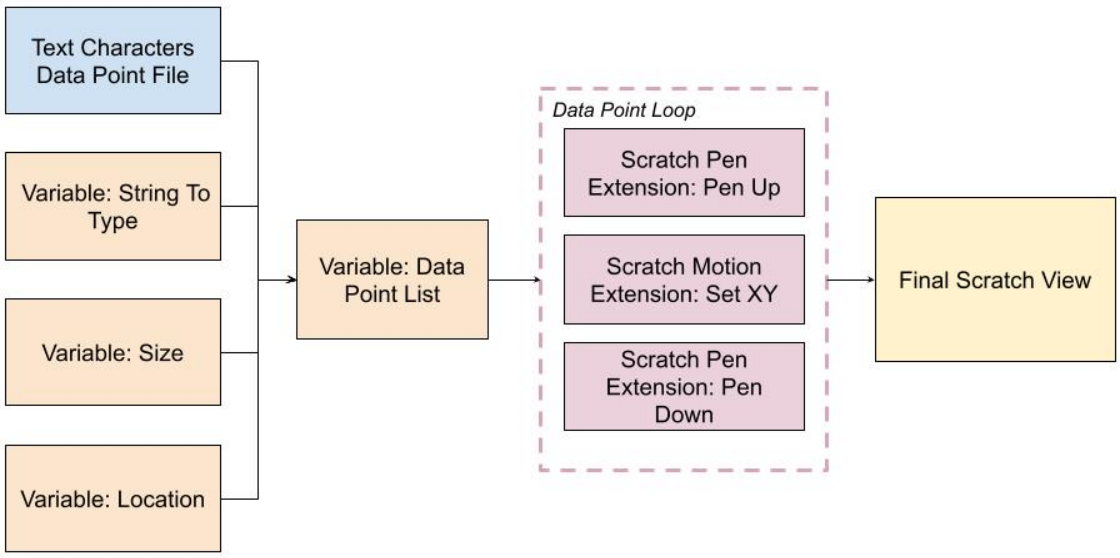


Figure 5-18: Text Rendering Backend System Diagram.

The text rendering utilizes a file that stores the names of different characters, and maps them to a list of normalized data points such that they are all the same

height, and the first point always starts at the bottom left corner of the character. The width of the character is also stored so that the code knows much space to skip before drawing the next character, in order to ensure uniform spacing between characters. This file is then used to call a function to write a string, such that the function receives a string to write, and for each character in the string, the data points that correspond are looked up, as well as the spacing. The function is also given a starting position, (x, y), as well as a size scaling factor. This is then used to create a full datapoint set that will then be traced with the Pen extension in order to create the string. From here, the pre-existing Scratch Pen extension is used, which has a penUp() and penDown() function, to decide whether the cursor will write or not. The code then uses the util.setXY() function such that for each character, the pen is put down starting at the initial (x, y) position, then uses setXY() for each data point in the list after scaling it by the scale factor. The code then makes sure to call penUp() again so there is no connection between letters. This is repeated for each letter, skipping by the appropriate amount for each letter to make room for the next one. This then creates the final string at the correct size and in the correct location.

5.7 Music Creation, Accompaniment, and Visualization Integration

Because music creation, accompaniment, and visualization all utilize the same data structures, some design thought needed to go into ensuring that all information would stay correctly updated. The goal was to complete as few updates as possible while still maintaining concurrent information. These three subsystems shared, volume, instrument, and note list information, so it was important to ensure that these variables remained consistent throughout.

The note list was shared between each of these libraries, as it was necessary to keep them all updated together so that visualizations would match the music and vice versa. It was also important to transfer knowledge about the instruments

and volumes chosen at any given point so that this information would also stay consistent throughout. As music creation updated music, this new data would be sent to visualization and accompaniment as the user called blocks from those sections. This also followed for any other transfer between block subsystems – the code did not update any other subsystem until the user called a block in a new system, and then it made sure to do all of the data transfer before the new system could make any changes to anything, in order to ensure it was working with the most updated states. Each time a block using note list was called, the original note list stored in music creation was sent along, so the other subsystems could update as necessary.

Chapter 6

Evaluation

The Machine Audition curriculum and accompanying Scratch blocks were used in two different middle school pilots. The goal was to evaluate our curriculum and Scratch blocks based on the original goals set for the project:

1. Create a Scratch extension for use in a Machine Audition Curriculum that can support signal analysis and visualization, and music creation and accompaniment.

2. Interface with Machine Audition Curriculum to create interactive, educational activities that can be done using the Scratch blocks created.

3. Support middle-school students in learning about machine audition.

In order to evaluate these goals, it was necessary to separately evaluate both the usability of the Scratch blocks on their own, as well as how well they facilitated the discussion around the machine audition curriculum. It was also important to ensure that students had learned about the signals properties that had been discussed, and that the Scratch blocks had enabled them to understand them better.

Two iterations of the curriculum were piloted to evaluate these goals – the first was a part of a FutureMakers workshop held in March 2021 over students' spring break over Zoom. This pilot had students ranging from 5th to 11th grade. A new iteration was then piloted in June with the Advanced Math and Science Academy

Charter School (AMSA), with two 8th grade classes. This chapter will describe the pilot methodologies, the evaluation assessments created to gauge the material being piloted, and the findings that came of both pilots.

6.1 Future Makers Spring Break Pilot

The Future Makers pilot occurred over one week, in 3 hour blocks each day. This was conducted entirely over Zoom, and consisted of students from 5th to 11th grade.

6.1.1 Methodology

The Future Makers pilot was fully remote and lasted a week. Each day consisted of two 90 minute sessions, separated by an hour lunch break. The youngest students were in 5th grade, while the oldest was in 11th grade. Each day only 2-3 students were in attendance, and no student was able to attend every day of the week. In that time, the pilot was able to cover Modules 1 and 2 of the curriculum over this week: Audition Overview and Signals (Pressure Waves). At the beginning of each module, students were given a pre-assessment to gauge their prior knowledge of the material. From here, the pilot followed a Zoom lecture format, with a screenshare of the slides for each module. This was accompanied by supplementary videos to explain more abstract concepts, as well as discussions every so often to help students test their understanding of the material thus far. At the end of each module, a post-assessment was given, which was the exact same questions as the pre-assessment, in order to compare what students had learned. It took about 2 days to get through Module 1, before the Wednesday session had to be cancelled due to low attendance. The students then were able to get through a portion of Module 2 on Thursday and Friday, but were not able to introduce any Scratch activities. Thus, the pre- and post-assessments were given for both modules, but only the amplitude and pitch portions of module 2 were taught, not harmonic frequencies. Furthermore, no student took both the pre- and post-assessment for the same module. A table detailing the schedule of this pilot can be seen in Table A.6.

The evaluation assessment used for this pilot can be seen in Figure B-6.

6.1.2 Results

As mentioned before, this pilot ended up being inconclusive. Not only were students in and out during the pilot due to other obligations, the group also had a much too wide age range to keep everyone engaged. Because the pilot ran over spring break, the program was fairly relaxed and led to poor attendance. It was discovered that the 5th graders were struggling with filling out the Google forms that had been provided, and were often needing much more time than anticipated just to fill them out.

Only three data points were gathered for the signals module pre-assessment of the curriculum. When asked the four questions in Figure B-6, students were also asked to give explanations for their answers, which were then used to determine if students had truly grasped the curriculum material. Students were able to get the amplitude question right 100%, and were also able to cite the height of the wave as the reason. However, the frequency question was answered correctly by only 2 of the 3 students, but both of those students were able to explain the reason why. The third student conflated amplitude and frequency, saying "The blue one is a little bit higher," as their reasoning for why they chose D4 instead of the correct answer, C5.

None of the students were able to answer the fundamental frequency question correctly. Two students wrote that they did not know the answer in the explanation box, while the third wrote "It's not intense,", for their choice, citing the height of the frequency value rather than looking for the lowest frequency on the X axis.

Finally, students were also not able to answer the question relating to higher harmonics, and being able to determine whether the two graphs represented instruments playing the same note. Students cited reasons such as "the bass has some missing notes," showing that they were not able to identify the properties of fundamental versus harmonic frequencies.

By the end of the module, the students that had answered the pre-assessment were not in attendance, and the students who were in attendance were not able to fill out the form. Thus, it was not possible to make any conclusions about the effectiveness

of this curriculum from the Future Makers pilot.

6.1.3 Future Maker Conclusions

This pilot resulted in a new understanding of the target audience for the curriculum. It was concluded that 5th graders were too young, as they lacked the ability to think about abstract concepts such as audio waves and signal processing. On the other end, it was determined that although the high schoolers were interested in the material, the way the lessons were presented was a bit too juvenile for them. The high schoolers also ended up spending a lot of time waiting, while the younger students took more time to complete the same activities. Thus, it was concluded that the optimal age range for this curriculum was 6th to 8th graders. This knowledge of a new age range allowed another pilot to be set up with the 8th grade class at AMSA.

As far as feedback for the actual curriculum content and Scratch, the results unfortunately did not yield much information. This was due to the fact that there were not enough data points to come to any general conclusions, and the students were also not able to get through enough of the material. No student was in attendance for both the pre- and post-assessment of any module, so no useful data could be gathered about the knowledge gained from the lessons. The students also were not able to get through any Scratch activities, which meant there was no feedback about their usability or educational ability.

6.2 AMSA Pilot

6.2.1 Methodology

The AMSA pilot was conducted over the course of 2 weeks. 2 different class sessions piloted the curriculum, both with the same teacher in a computer science class at AMSA. Each class met twice for the pilot, for a total of 2.5 hours. During the first session, the students filled out a pre-assessment and then went through Module 1 of the curriculum. They also went through the early stages of Module 2, so that the

next lesson could be interactive Scratch activities instead of lecturing. During the second session, students were able to complete a more interactive session in which they went through the first 2 Scratch activities that had been designed. This allowed for a chance to assess the usability of the blocks, as well as how much they impacted students' learning. The students then completed the post-assessment for modules 1 and 2. This pilot resulted in 11 valid data points, all of which completed both the pre- and post- assessments for both modules.

A schedule for this pilot can be seen in Table A.7.

6.2.2 Results

In the AMSA pilot, significantly more material and activities were covered compared to the previous iteration. This pilot provided valid data points to assess the ability of the curriculum to teach students about signals and machine audition. First, the students were able to cover the entirety of Modules 1 and 2, and user test some of the Scratch activities. A significant amount of data was also collected from the students, and everyone filled out the pre- and post-assessments given, resulting in valid data points for comparison. A full table of student correctness scores over the course of the pilot can be found in Table A.8.

Amplitude

The assessment questions for amplitude can be seen in Figure B-7.

In the first pre-assessment amplitude question, the two signals have the same frequency, and only differ in amplitude. 18 students, or 78% of the class, were correctly able to identify the higher amplitude signal. All of the students that got the question correct were able to identify that this was due to the height of the waveform. Of the students who answered incorrectly, 3 conflated amplitude and frequency, incorrectly citing higher frequency contributing to higher volume. This was incorrect both due to the conflation of the properties, as well as the fact that the signals were both the same frequency. The remainder of the students explained that they had guessed. In

the same post-assessment amplitude question, all 23 students were correctly able to identify the higher volume signal. They were all further able to explain the property that contributed to this in their answer.

In the second pre-assessment question, the correct higher volume signal also had a lower frequency. This question was designed to catch students' confusion between amplitude and frequency signal properties. In the pre-assessment, 14 students, or 60% of the class, were correctly able to identify the higher amplitude signal. Every student who got this question wrong cited the higher frequency contributing to higher volume. In the post-assessment, all 23 students, or 100% of the class, got this question correct. The students also all correctly cited the amplitude as contributing to volume. This was a significant discovery, showing that over the course of the curriculum, students were successfully able to separate the amplitude and frequency properties of a signal, and understand that just the height of a wave contributed to the volume of the music.

Frequency

The assessment questions for frequency can be seen in Figure B-8.

In the pre-assessment for frequency, the first question followed a similar structure to the first question in amplitude. These two signals did not differ in amplitude, and only differed in frequency. Initially, 19 students, or 82% of the class, were able to answer this question correctly. Of the students who answered incorrectly, 3 had the relationship backwards, citing longer wavelengths to mean higher frequency. The remaining student did not know. In the same post-assessment question, 100% of the class was able to answer correctly. Furthermore, every student was able to correctly explain that this was due to the shorter period of the wave.

In the second pre-assessment question, the signals were deliberately designed such that one signal had a higher frequency, but lower amplitude, and vice versa. This was again meant to test for conflation of signal properties. Initially, 20 students, or 86% of the class, answered this correctly. All of the students who were incorrect cited the wrong property, claiming that the higher amplitude meant that the signal was higher frequency. In the post-assessment, again, 100% of the class was able

to answer correctly and cite the correct signal property as explanation. From this, it was possible to again conclude that the frequency module of the curriculum was successfully able to teach students how to identify frequency properties of a signal.

Harmonic Frequencies

The assessment question for harmonic frequencies can be seen in Figure B-9.

Both of the frequency plots shown are playing the same note, as can be seen by the matching fundamental frequency value at 400. In the pre-assessment for this question, however, only 6 students, or 26% of the class, were able to correctly answer. Of the students who were correct, only 2 cited the fundamental frequency matches to be the reason. The rest of the students incorrectly cited the fact that there was the same number of frequencies, or that the frequency values all matched. Of the students who answered incorrectly, most said that they were not the same note because the graphs were not exactly the same, and a handful of students did not know.

In the post-assessment question, there was not much improvement. Only 7 students, or 30%, answered correctly. Of those students, only 3 were able to cite the fundamental frequency, while the others again cited the location of all of the frequencies present. The students who got the question wrong again claimed this was due to the fact that it was not the exact same graph.

Unfortunately, this concluded that the timbre and higher harmonics module of the curriculum was not successful. By the end of the pilot, students were not able to identify the difference between fundamental and harmonic frequencies visually. Although they were able to think critically about the placement of the frequencies, they could not determine that the lowest frequency was related to pitch, while the higher harmonics were related to timbre.

Scratch Interactions

The AMSA pilot allowed for the testing of 2 Scratch activities with the students: Know Your Dynamics and Making Melodies. This gave the opportunity to see how usable the blocks were, as well as how much they were able to aid in students' un-

derstanding of the curriculum. As the students worked in Scratch, it was possible to see that they could easily begin using the blocks and understand their functions, and it was not found that users were getting stuck on the actual implementation of their Scratch projects. In the Know Your Dynamics activity, students worked in groups to make their secret volume-varying music, before showing students their visualization and having students recreate it. The students split up into 3 groups of 4 per section. In both sections it was possible to see that every student was able to guess the correct amplitude order relatively. For example, if the original signal started at fortissimo and then went down to mezzo-piano, even if students did not guess these exact volumes, they were still able to conclude that the volume had lowered relatively. This happened across the board, and it was observed that there was a 100% accuracy rate for relative volume answers. The students were also all able to create their music in just a few minutes, and come up with their answers in a similar amount of time. This showed that the activity was easy to use, and did not get students stuck trying to figure out the workflow.

Making Melodies was a bit more difficult for the students. It is believed that this activity introduced too many music terms, and needed to be much simpler. This activity did allow students the opportunity to create music, and they did not struggle to utilize the blocks. They were given a few minutes per music creation, and were able to create something in that time. However, students really struggled to understand major and minor scales, as well as the intervals that were discussed in the lesson. This was where all of the questions came from, showing that the biggest challenge with this activity was the music terminology. However, students were able to create and analyze their music, and make sense of the relative changes in frequency in the waveform visualization as they modified their music.

6.2.3 Conclusions

Firstly, the second pilot showed definitively that this curriculum is long and difficult to get through in shorter periods of time. Students were only able to get through Modules 1 and 2 in 3 hours, and were not able to do all of the associated Scratch

activities in that time. In the future, it would be necessary to allocate 3-4 times this amount of time to the curriculum in order to possibly get through the entire curriculum. This might mean that the optimal curriculum timeline would be the 3 hours a day schedule that was attempted in FutureMakers, but with an older group of students that would progress through the curriculum faster. Alternatively, this curriculum may need to be broken up into smaller sections, perhaps first covering signals in one curriculum, before a separate curriculum that would then cover the machine learning portions of this curriculum.

From the material students were able to get through, the section that needed rethinking was the connection from timbre to higher harmonics. The instructors were able to go over the baseline information, but did not end up intuitively explaining to the students how to determine the difference between fundamental frequencies and harmonic frequencies. This might have been partially due to not doing the Scratch activity that related to it, so in the future more data would be needed from the activity before being able to definitively conclude what needed to change in the lesson plan.

6.3 Summary

6.3.1 Findings

During these two pilots, it was possible to determine that the optimal target audience for the curriculum was 6th to 8th graders. It was also found that students were able to use the Scratch blocks effectively, and were able to understand their function intuitively. They were able to engage in the activities provided for them in Scratch with ease, and were clearly utilizing the lessons presented in their discussions around the Scratch blocks. The students also displayed a newfound strong understanding of the relationship between amplitude and volume, as well as fundamental frequency and pitch, by the end of the curriculum. They also remained engaged and in discussions every time they were asked questions, and generally were answering correctly when called on. This showed that the curriculum itself was maintaining student's

engagement, and they were showing an interest in the way the content was being presented.

6.3.2 Limitations

One of the main elements of Module 2 that students seemed unable to grasp after the curriculum pilots was timbre and higher harmonics. Although students were able to understand that timbre was caused by the higher harmonics of a signal, they were unable to differentiate between fundamental and harmonic frequencies on a Fourier transform plot or a spectrogram. Furthermore, students did not have as much time as needed with curriculum. This meant it was not possible to pilot some of the Scratch activities, and some blocks unfortunately went untested.

Chapter 7

Future Work

The limitations of COVID as well as the 2 semester timeline made it clear that more work could have been done to improve the Machine Audition curriculum and its Scratch blocks. It was also discovered through further data in the pilots that there are various areas where the Scratch blocks could be made more intuitive or more engaging, or simply have more widespread use cases. This chapter breaks down all the future work that can be done primarily on the Scratch blocks themselves as a tool for future iterations of the machine audition curriculum as well as for independent creative use.

7.1 Music Creation

As is, the machine audition Scratch blocks have some limitations that make it difficult to create more complicated music. First, in future iterations the Scratch blocks should be extended to support simultaneous note playing. Currently, the backend player is only able to handle one note at a time, and running multiple players at once introduces concurrency issues that stop the player entirely. It is important to modify this such that multiple players can be running at once in order to play multiple notes at the same time, as this will significantly expand the variety of music that students can create with these Scratch blocks. This would also pave the way for the inclusion of chords as input, such that students could simply use a block that allowed them to

give a multi-note chord name, and they could include that in their music without having to choose each individual note of the chord. That chord could then be played with all notes simultaneously with the support of a threadsafe player. Furthermore, it would be important to support multiple instruments playing at the same time as well, so that each of these music parts can be played by different instruments in order to introduce more full orchestra music for students to create. Currently, if two different instruments are selected concurrently, the block that was selected slightly later will be chosen for both.

Furthermore, the current method of adding notes to a song is somewhat tedious, as it requires a separate Scratch block for each note that a student wants to add. It would be interesting and engaging to include a recording feature, where students could play a virtual piano while it was recording their melodies, and this would then be transcribed in the backend into a note list, or turned into a series of Scratch blocks that encoded the same music. This way, students could spend less time doing the tedious work of dragging many blocks in order to create complicated music, and spend more time working on the music itself.

Finally, the beat choices for the music creation blocks are somewhat limited currently. In the future, the beat choices could be expanded to include notes that are shorter than 1/8th notes in order to allow for more complex, fast-paced music. The music creation blocks could also be extended to allow students to input rests, such that there can be pauses in their music.

7.2 Music Analysis

Currently, the Scratch blocks created can execute a simplified Fourier transform, and plot sheet music, frequencies, spectrograms, and waveforms. All of these have all the information necessary for students to learn the machine audition material, but can be improved to add more support for more complicated music and education. First, it would be helpful if the Fourier transform functionality was made more efficient such that it could analyze longer pieces of music in a short amount of time. The

spectrogram can also be improved by giving students the ability to set the window and step size for each spectrogram iteration, in order to allow them to have better control over the time and frequency resolution of the resultant spectrogram. This would be meant for more advanced students who have a better understanding of signal properties and can use that information to generate a high resolution spectrogram.

Finally, it would be ideal if students could compare any two files that they give, and not just the default ones that were given to them. As is, students are given a choice of mystery files to ask which is louder, which is higher pitched, and what instrument is being played in both. However, if students could do this with their own music files, they would be more engaged in the content and the results to these questions, and they would be able to compare a wider variety of music in order to better understand how these properties change the sound of music.

7.3 Music Visualization

The music visualization blocks are currently able to produce clean, easy to read visualizations that allow students to understand the relative differences between different pieces of music. However, this can be improved to encode more information.

First, the labeling on all of the plots (waveform, Fourier Transform, and spectrogram) can be improved. As is, the labeling is simply relative, so students can see where higher or lower frequencies exist in their music, but are unable to see the actual frequency values. It would be important for more complicated music creation and signals understanding to include actual frequency values in Hz on the graphs, and also include time labels in seconds on the spectrogram and waveform visualizations. The plots could also use better coloring, as will be discussed in more detail for each specific visualization.

Second, the text rendering itself still has a ways to go before it is fully robust. Currently, text is rendered by converting a string into a series of data points that are stored in a file for each letter in the string. Thus, if a character is not in the file yet, it cannot be rendered and will be skipped. The code should be modified

to first stop failing silently, and second to be more robust and support all possible characters. Text rendering also currently uses only one font, which could be modified with a more robust backend. This text rendering is functional and can be used to write text anywhere on screen at any size, but it cannot change fonts, be rotated, or include other characters that have not been manually entered into the backend. Given more time, the text rendering should be written in a more robust way that allows for easy changes in font, rotation, and addition of characters.

It would be helpful if overall the visualizations were able to work more in real-time. It would be good for students to be able to see the visualizations dynamically change as they changed instrument type, added notes, or adjusted the volume of their music, so they could see how these things changed their music visualizations in different ways. Currently, the visualization only updates after it plays the music all over again, which can be a little slow, especially if the music is longer. Having it update dynamically would help students see in real-time how their settings change their music, and make the entire process much smoother and faster.

7.3.1 Waveform

The waveform visualization could be made more robust by generating more than 4 colors to plot notes in. Currently, because the visualization is hard to read if more than 4 notes are included, there is no support for longer note sequences. This support should be added regardless, or the block should actively limit users from adding more than 4 notes, which it does not currently.

7.3.2 Fourier Transform

Currently, there is no color coding in the Fourier transform visualization, though it would be beneficial to users to do this. If the transform visualization was color coded such that fundamental notes were paired with their harmonics in the same color, or such that fundamentals were labelled as one color while higher harmonics were another, users might be able to better understand the Fourier transform graph,

especially since it does not retain any time information about the signal, making it very difficult to understand once multiple notes are added to it.

7.3.3 Spectrogram

The spectrogram visualization as is runs very smoothly, but could be improved with some more feedback from the user. For example, offering window size and step size as adjustable parameters for the user would allow them to explore the time and frequency resolution tradeoff. Furthermore, the spectrogram should have more specific labelling, showing users which frequencies are fundamental vs. harmonic.

7.3.4 Sheet Music

The sheet music visualization needs more support for the kinds of notes it can convey. First, the sheet music currently defaults to common time, and only allows students to choose from a set list of note durations in order to limit the necessary visualization support. In future iterations, this should be modified to include smaller note durations, and support this in the visualization. It would also be helpful if users could select a time signature and key signature, and these could be reflected in the sheet music visualization. Finally, the sheet music should be modified such that it will put notes on the appropriate clef according to the instrument that is being played, rather than the current default of placing a note where it would be written for a grand piano.

7.4 Music Accompaniment

The current machine accompaniment module of the Scratch blocks simply plugs in a Google Magenta backend in order to either generate, complete, or accompany music that the user has given already. This could be improved in a few ways. First, Google Magenta is currently only able to use a piano to create the new music. It would be helpful to determine how to interface with the library in order to change the instrument to reflect the setting the user had selected.

Furthermore, the Magenta blocks currently take a substantial amount of time to run due to the machine learning backend. Figuring out a way to optimize this such that it had a lower latency would be very helpful in ensuring the process of music generation was as smooth and efficient as possible.

7.5 Backend Architecture

The Scratch blocks created added some important functionality to Scratch as a whole. In the future, it would be much easier to modularize this code such that other extensions could simply use the functions created to perform the same tasks as the machine audition blocks. First, text rendering can now be used everywhere in Scratch, but is still located within the code for the machine audition extension. In the future, the text functionality should be moved into a new file such that other extensions can call it as helper functions for use in other extensions. Second, the plotting functionality should also be moved to a separate file such that other extensions can use plotting in their own code easily. Finally, the sheet music visualization can be used with any music now, provided that a helper function is created to refactor the music data structure into a format that my code can understand. In the future, plotting, sheet music, and text rendering can all be converted into backend code that all extensions can use, as these are the three main contributions that have been made to Scratch as a whole.

7.6 Further Pilots

Finally, it would be important to plan more pilots in the future to cover more of the Scratch blocks that were created. It would be helpful to try out these blocks in more curriculum pilots, as it was only possible to test in one environment that gave good data. It would be important to test this on a wider audience of students with a more diverse set of skills, in order to ensure that these blocks are actually kid-friendly and self-explanatory enough to be used by any middle school student,

regardless of Scratch or signals experience. More user testing is required in order to determine whether these blocks contribute to students' learning in the machine audition curriculum, especially in the timbre module, as well as the final machine audition capstone projects.

7.7 Creative Use of Scratch Blocks

In the pilots that did run, students only used the blocks in structured activities that were premade. Thus, the most the students were able to change was selecting a new instrument, a new volume, or adding extra notes to the music. This was helpful for assessing the blocks' ability to assist in students' learning, but it did not provide much insight into the usability of the blocks on their own. In the future, piloting a more creative workshop where students could more freely use the blocks would be much more helpful for understanding what the actual variety of creation these blocks offer. This would also enable more data collection to understand what parts of the Scratch block design is non-intuitive or tedious, so the blocks could be improved.

Chapter 8

Conclusion

8.1 Contributions

This paper shares the design process behind a machine audition curriculum targeted at middle school students, with the goal of educating students on the differences between human and machine audition. The curriculum focuses mainly on computational signal processing and educating students on signal properties and how they impact music, as well as the human-AI collaboration that can help create new pieces of music which then can be analyzed as well.

A new Scratch extension was designed and developed that allows users to create, analyze, and computationally generate and accompany music. This involved the ability to create music and adjust its volume and instrument type over time, visualize these signals in many different formats to understand how the properties of signals changed the way music sounds, and the ability to easily utilize Google Magenta to use machine learning to generate and accompany music.

8.1.1 Scratch Contributions

Main additions to Scratch include the ability to plot time-domain graphs, as well as Fourier transform magnitudes, spectrograms, and sheet music. These visualization blocks can be used to plot any music file in order to better understand its properties.

This functionality was able to use the Scratch Pen extension in order to easily create plots, and can now be used across Scratch as a whole to plot any dataset in the Scratch window. The sheet music visualization can also be used with any data structures that are properly formatted to represent music notes over time. Furthermore, Scratch functionality was added that allows text rendering at any location on screen. This allows for the design of future educational blocks that can create and label diagrams for use in other curricula. Finally, a simplified digital signal processing functionality was added to Scratch – users can now re-purpose these blocks in order to perform an Fourier transform, or a short-time Fourier transform, and extract the frequency content of a signal, either across the waveform in its entirety, or over time. This can then be used with other code blocks as information or to be plotted.

8.1.2 AI Education Contributions

This project also shows significant progress made in tuning the curriculum to be better suited to students. It was possible to determine that the curriculum was best suited to students in 6th to 8th grade, and that any younger will not be able to think about the abstract concept of signals as easily, while older students will not be as engaged by the character designs and the lecture presentations that have been designed.

It was also possible to give students the opportunity to play with the Scratch blocks on their own in new and creative ways, and see that students were able to easily use the Scratch blocks to generate music successfully, and they were also able to visualize their music and understand what the plots were saying. It was also found that the Scratch activities were able to facilitate discussion among the students as they interacted with each other in groups, helping them solidify their understanding of the topics at hand as they worked.

Finally, students' learning from the curriculum was tested. It was found that the curriculum as is currently does a good job of helping students understand frequency and amplitude as signal properties, as well as how they impact the pitch and volume of a sound, respectively. However, there was a point of improvement in the higher

harmonics and timbre lesson. Although students were able to realize that higher harmonics impacted timbre, they were not able to identify when two different notes were being played, versus when it was simply two instruments. This meant the difference between fundamental frequencies and harmonic frequencies had not been explained visually, and would need to emphasize this in spectrogram visualizations in the future.

This project provided a powerful new Scratch extension capable of signal processing, music creation, signal visualization, and computational music generation through Google Magenta. This project was able to contribute to the Media Lab's current AI Education initiative, and provide an engaging, middle school curriculum that students definitively learned from.

Appendix A

Tables

Block Name	**Functionality**
Music Comparison	
Volume Comparison	Takes in two single-note music files, and determines the amplitude (volume) of both
Pitch Comparison	Takes in two single-note music files, and determines the fundamental frequency (pitch) of both
Timbre Comparison	Takes in two single-note music files, and determines the harmonic frequencies (timbre) of both
Music Creation	
Create New Sound File	Initializes empty music storage data structure that the user can iteratively add notes to
Set Amplitude	Takes in a volume in dB, stores this in amplitude variable in order to add a later note to the sound file
Set Frequency	Takes in a frequency in Hz, stores this in frequency variable in order to add later note to the sound file
Set Instrument	Takes in an instrument name, stores this in instrument variable in order to add a note to the sound file later
Set Duration	Takes in duration in seconds, stores in duration variable in order to add a note to the sound file later
Add Note	Takes in amplitude, frequency, duration, and instrument variables (if all filled in) and adds a note with the given specifications to the sound file
Music Visualization	
Time-Domain Signal	Takes in an audio file, plots signal in time domain and labels important features of the signal (amplitude, fundamental frequency, etc.)
Fourier Transform	Takes in an audio file or its fast Fourier transform, plots discrete Fourier transform and labels important aspects (fundamental frequency vs harmonic frequencies, amplitudes, etc.)
Spectrogram	Takes in an audio file or its short-time fourier transform, plots frequency over time and labels important aspects (fundamental frequencies vs harmonics, amplitudes, durations, etc.)
Sheet Music	Takes in an audio file or short-time fourier transform, plots fundamental frequency over time with sheet music visual and labels note names/corresponding frequencies
Music Accompaniment	
Accompany Music	Takes in a music file and a genre, uses machine learning to generate an accompaniment based on music theory for the specified genre, returns accompaniment signal

Table A.1: Initial Design of Machine Audition Scratch Blocks

Block Name	**Input**	**Output**	**Functionality**
Visualize Waveform	None	Time-domain waveform visualization	Displays time-domain signal visualization of the notes the user has added so far
Visualize Frequencies	None	Fourier Transform magnitude visualization	Displays frequency content of notes added so far
Visualize Frequencies Over Time	None	Spectrogram visualization	Displays frequency information over time for notes added so far
Visualize Sheet Music	None	Sheet music visualization	Displays sheet music visualization of notes added so far

Table A.2: Music Visualization Scratch Blocks

Block Name	**Input**	**Output**	**Functionality**
Reset Music	None	None	Clears music list data structure
Set Instrument	Instrument selection from dropdown menu	None	Updates instrument variable for use in future notes
Set Volume	None	None	Updates volume variable for use in future notes
Play Note For Beats	Note selection from virtual piano, Beats from dropdown menu	Playback of note selected	Adds note, duration, instrument, and volume information to music data structure
Get Volume	None	Volume displayed on screen	Returns last volume selection
Get Instrument	None	Instrument displayed on screen	Returns last instrument selection

Table A.3: Music Creation Scratch Blocks

Block Name	**Input**	**Output**	**Functionality**
Play Mystery File	Number from dropdown menu	Playback of selected file	Plays sound file corresponding to selected number
Compare Mystery 1 and Mystery 2	2 numbers from dropdown menus	None	Updates higher, louder, instrument variables with answers for selected files
Higher	None	Higher file displayed on screen	Returns which of the last two files played had a higher pitch
Louder	None	Louder file displayed on screen	Returns which of the last two files played had a louder volume
Instrument 1	None	Instrument 1 displayed on screen	Returns the instrument played in first file selected
Instrument 2	None	Instrument 2 displayed on screen	Returns the instrument played in second file selected

Table A.4: Music Comparison Scratch Blocks

Block Name	**Input**	**Output**	**Functionality**
Generate Music with Magenta	None	Audio playback of generated music	Sends request to Magenta to randomly generate music, and stores it as user's created music before playing it
Complete Music with Magenta	Current music, Duration and tempo of new music	Audio playback of completed music	Sends request to Magenta to create music that fits with user's current music with the duration and tempo specified, adds it to user's music, and plays it

Table A.5: Music Accompaniment Scratch Blocks

Monday	**Tuesday**	**Wednesday**	**Thursday**	**Friday**
Module 1: Why we need audition	Module 1: Hearing and listening machines	CANCELLED	Module 2: What is sound?	Module 2: Amplitude and Frequency

Table A.6: Future Makers Pilot Schedule

Class Section	**Day 1**	**Day 2**
1	Module 1: Why we need audition, Hearing and Listening Machines. Module 2: What is Sound?	Module 2: Amplitude and Frequency. Scratch Activities: Know Your Dynamics, Making Melodies.
2	Module 1: Why we need audition, Hearing and Listening Machines.	Module 2: What is Sound?, Amplitude and Frequency. Scratch Activities: Know Your Dynamics, Making Melodies

Table A.7: AMSA Pilot Schedule

Assessment	**Amplitude 1**	**Amplitude 2**	**Frequency 1**	**Frequency 2**	**Harmonics**
Pre	18 (78%)	14 (60%)	19 (82%)	20 (86%)	6 (26%)
Post	23 (100%)	23 (100%)	23 (100%)	23 (100%)	7 (30%)

Table A.8: AMSA Pre- and Post- Assessment Correctness Data

Appendix B

Figures

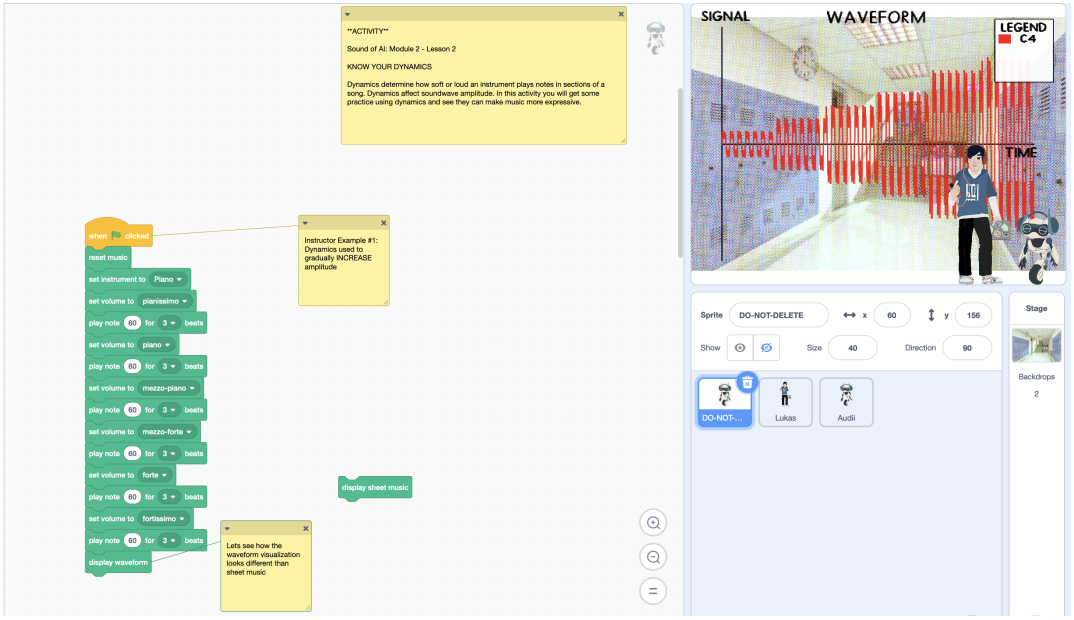


Figure B-1: Scratch Activity: Know Your Dynamics

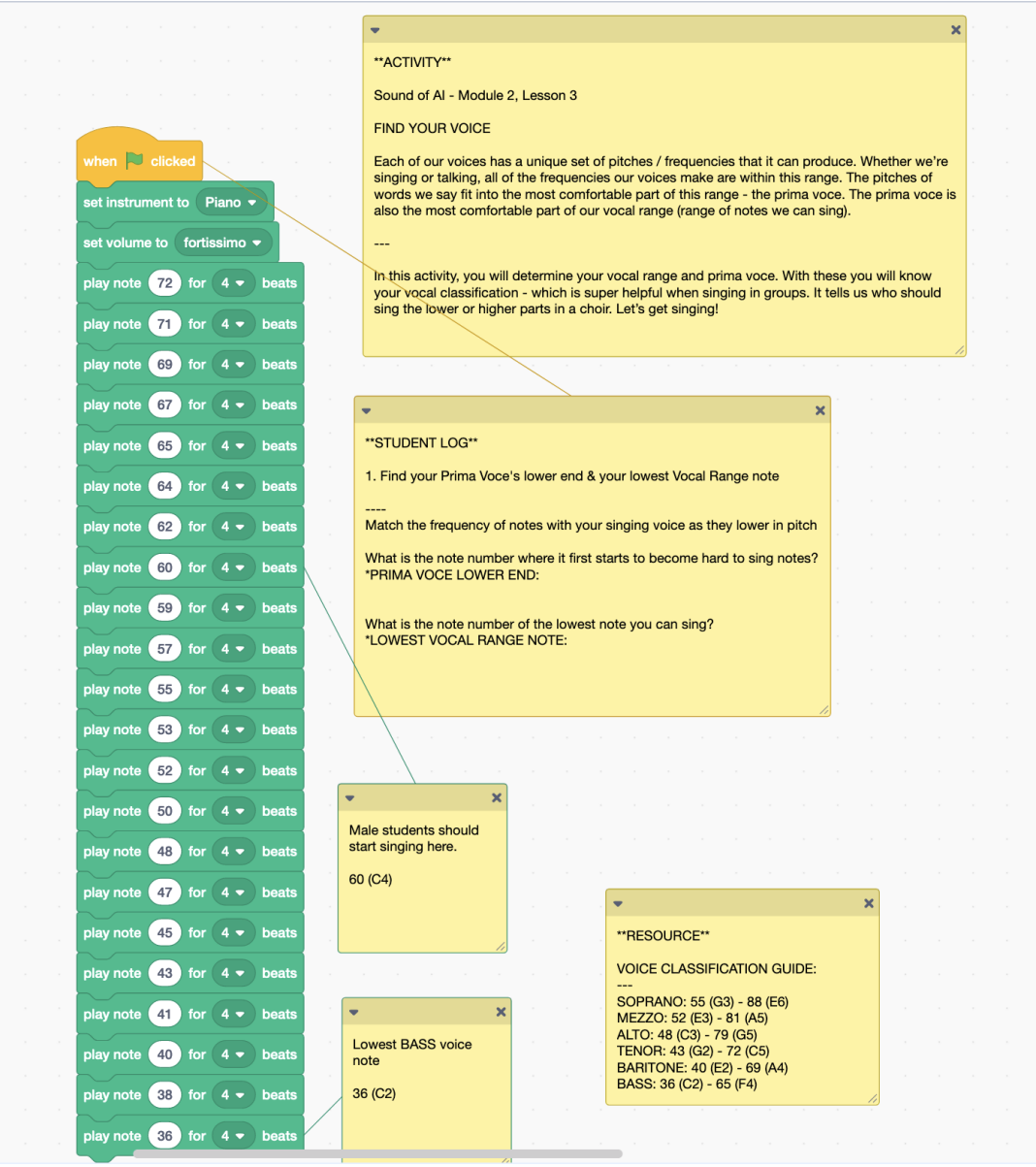


Figure B-2: Scratch Activity: Find Your Voice

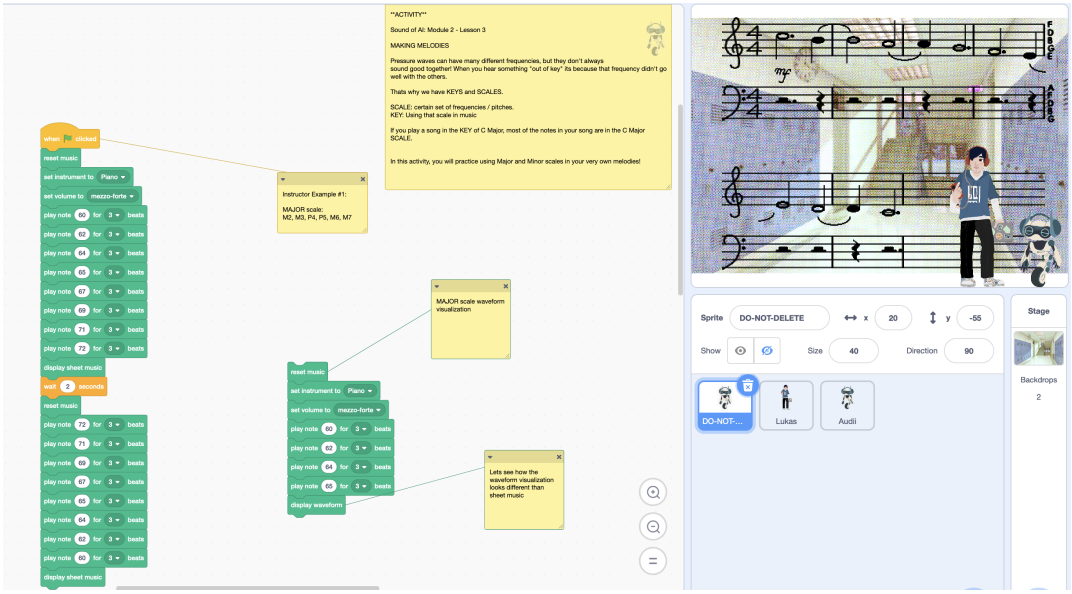


Figure B-3: Scratch Activity: Making Melodies

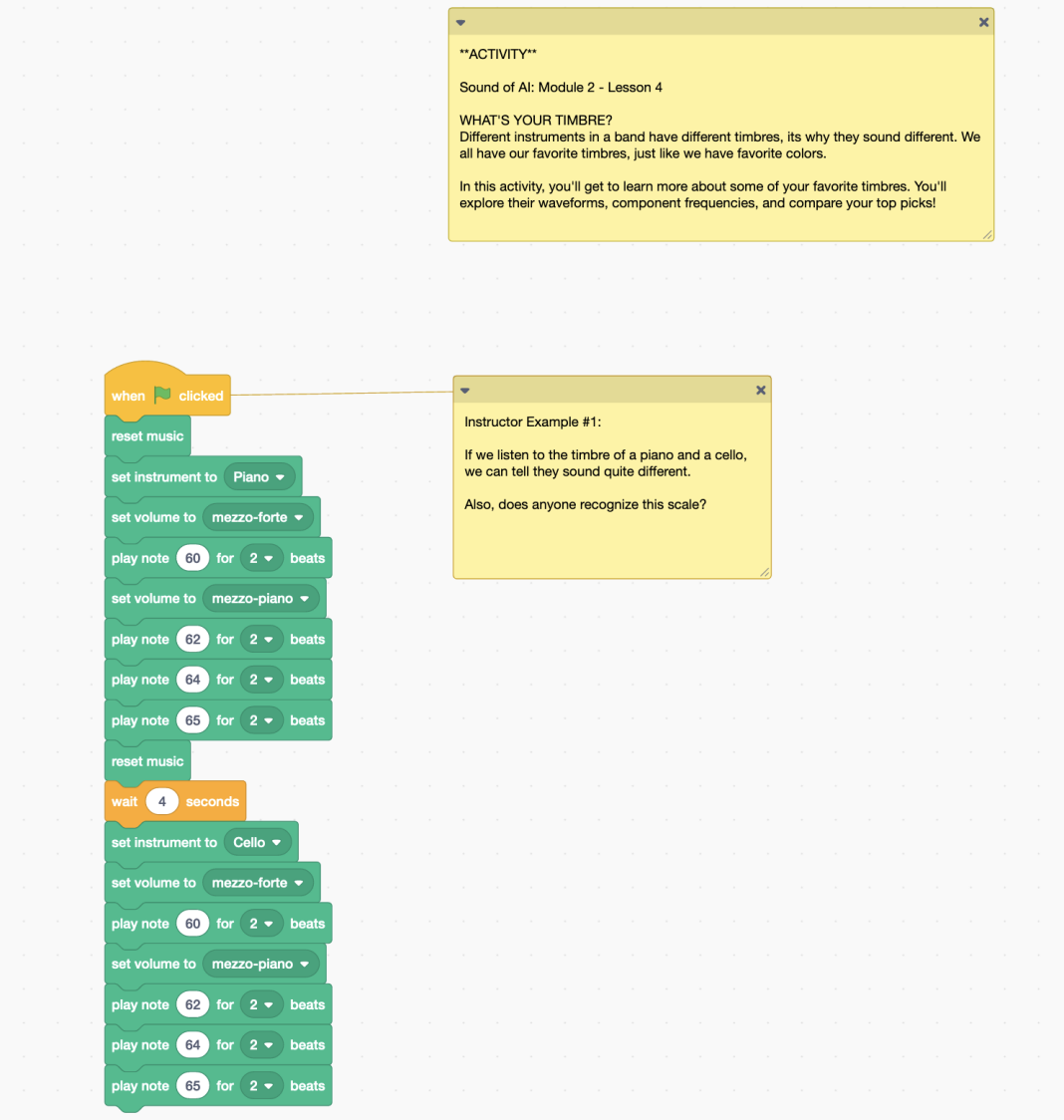


Figure B-4: Scratch Activity: What's Your Timbre?

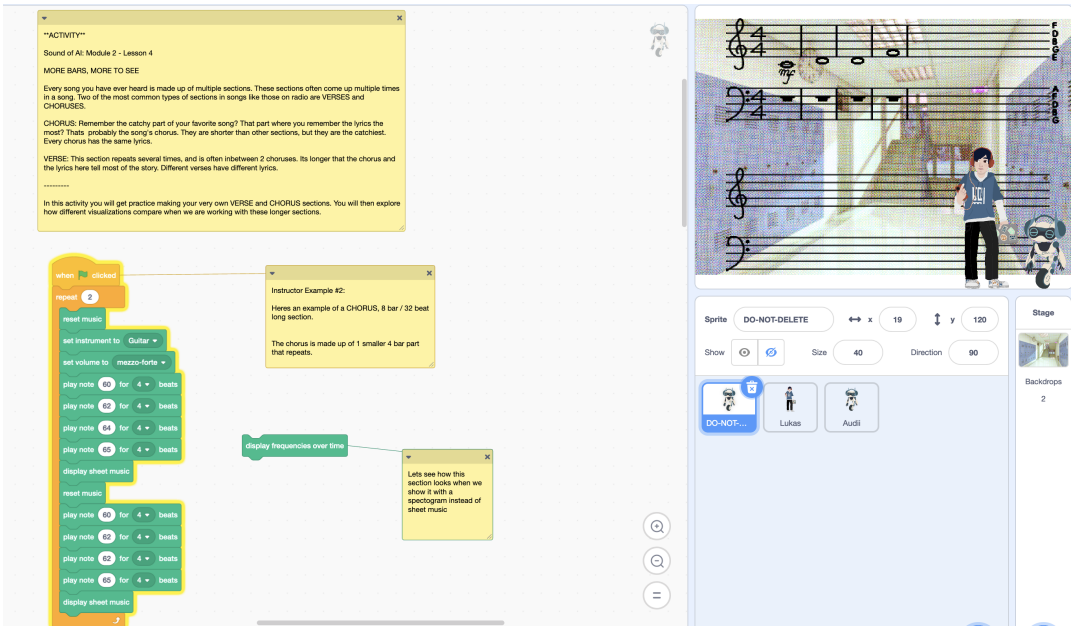


Figure B-5: Scratch Activity: More Bars, More To See

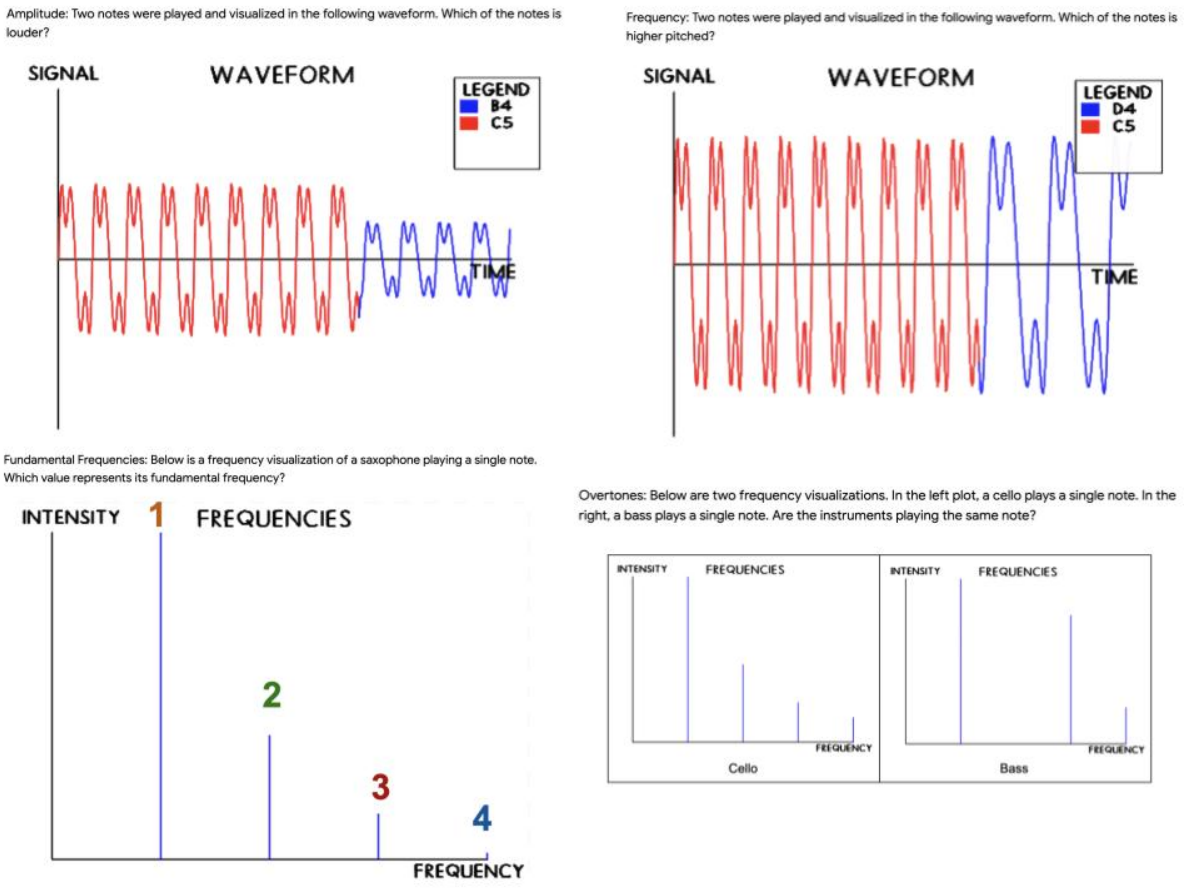


Figure B-6: Future Maker Signals Assessment Questions.

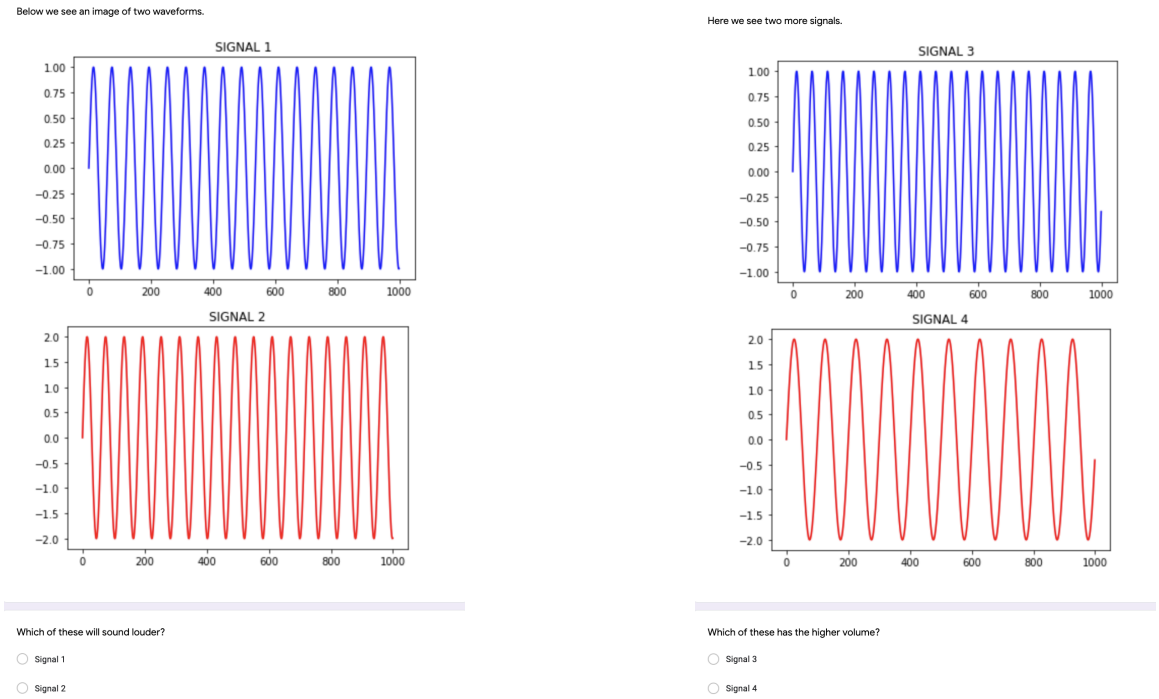


Figure B-7: AMSA Pilot Amplitude Questions.

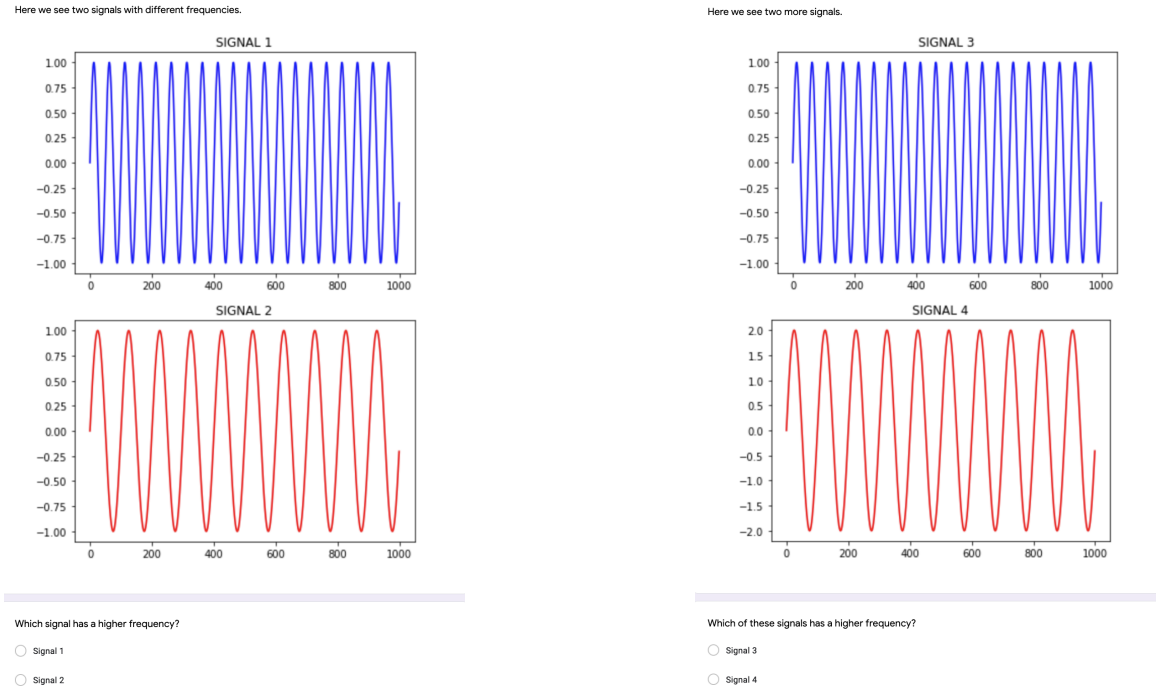
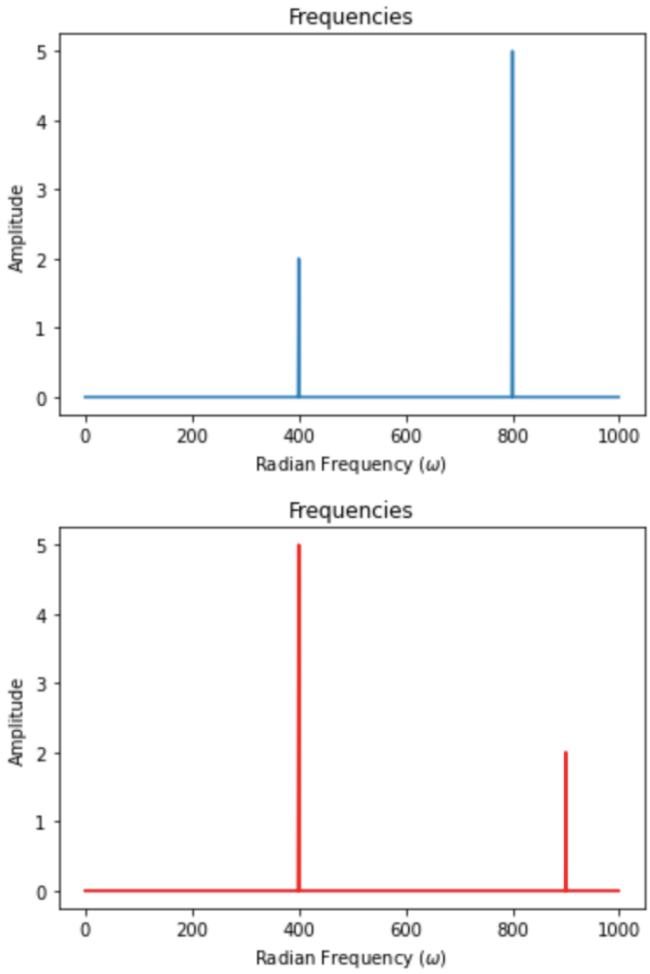


Figure B-8: AMSA Pilot Frequency Questions.

Below we see the frequencies present in two instruments each playing a single note.



Are the two instruments playing the same note?

- Yes
- No

Figure B-9: AMSA Pilot Harmonic Frequency Question.

Appendix C

Links

C.1 Live Activities

1. Know Your Dynamics: https://mitmedialab.github.io/prg-extension-boilerplate/machineaudition/?project=https://nadah09.github.io/M2.L2%20-%20Know%20Your%20Dynamics.sb3

2. Making Melodies: https://mitmedialab.github.io/prg-extension-boilerplate/machineaudition/?project=https://nadah09.github.io/M2.L3%20-%20Making%20Melodies.sb3

3. Find Your Voice: https://mitmedialab.github.io/prg-extension-boilerplate/machineaudition/?project=https://nadah09.github.io/M2.L3%20-%20Find%20your%20Voice.sb3

4. What's Your Timbre?: https://mitmedialab.github.io/prg-extension-boilerplate/machineaudition/?project=https://nadah09.github.io/M2.L4%20-%20Whats%20your%20Timbre.sb3

5. More Bars, More To See: https://mitmedialab.github.io/prg-extension-boilerplate/machineaudition/?project=https://nadah09.github.io/M2.L4%20-%20More%20Bars,%20More%20to%20See.sb3

C.2 Video Walkthroughs

1. Know Your Dynamics: https://drive.google.com/file/d/1IYJ8JUXtP5uMKevGGpXBpQTZBwEddEq0/view?usp=sharing

2. Making Melodies: https://drive.google.com/file/d/1dCJNuE4_VzQ33VkrIsfHNXOmVvf8AyPb/view?usp=sharing

3. Find Your Voice: https://drive.google.com/file/d/1Dvm9LtAETIZABuRVQDS5LvO0HNMS--EH/view?usp=sharing

4. What's Your Timbre?: https://drive.google.com/file/d/1EJwDexuvYBTAilmGhTsFeoYsrYJw4LuN/view?usp=sharing

5. More Bars, More To See: https://drive.google.com/file/d/1_vs22ArHQy_HWd4R8wJxZXAUuisaFJzM/view?usp=sharing

Bibliography

[1] Google Magenta Music. Documentation. https://magenta.github.io/magenta-js/music/index.html.

[2] MmTss. Code. https://github.com/andrewhao/mmtss.

[3] PyTorch. Documentation. https://pytorch.org/docs/stable/index.html.

[4] scipy.signal. Documentation. https://docs.scipy.org/doc/scipy/reference/signal.html.

[5] Scratch. About. https://scratch.mit.edu/about.

[6] TensorFlow for JavaScript. https://www.tensorflow.org/js.

[7] Chris Donahue, Julian McAuley, and Miller Puckette. Adversarial audio synthesis. In *ICLR*, 2019.

[8] Brian Jordan, Nisha Devasia, Jenna Hong, Randi Williams, and Cynthia Breazeal. Poseblocks: A toolkit for creating (and dancing) with AI. In *Thirty-Fifth AAAI Conference on Artificial Intelligence, AAAI 2021, Thirty-Third Conference on Innovative Applications of Artificial Intelligence, IAAI 2021, The Eleventh Symposium on Educational Advances in Artificial Intelligence, EAAI 2021, Virtual Event, February 2-9, 2021*, pages 15551–15559. AAAI Press, 2021.