A Web-based Audio Visualizer

Fatemeh Abbasi Razgaleh

Contents

1 Task 1: Selecting Audio Features

Sound No.	Meyda Audio Features	Justification
Sound 1	RMS Energy	To measure the loudness over time
	Chroma	Since the sound is extremely harmonic, it would be fascinating to analyze the harmonic content such as chords, or to detect the keys
	Power Spectrum	To emphasize the differences between the frequency bins since they sound similar but are different
Sound 2	Spectral Centroid	Since the sound changes drastically over time (brightness)
	RMS Energy	To measure the loudness over time
	Spectral Flux	Since the sound changes so drastically, it would be interesting to measure how fast the spectrum of the signal changes
Sound 3	Spectral Centroid	Since the sound changes drastically over time (brightness)
	RMS Energy	To measure the loudness over time
	Spectral Flux	Since the sound changes so drastically, it would be interesting to measure how fast the spectrum of the signal changes

2 Task 2: Web-based Audio Visualizer

2.1 Introduction

This project is a web-based audio visualizer that uses p5.js and Meyda libraries to visualize the audio features. It also includes a speech recognition feature that allows the user set the background by saying the name of the color they prefer. Here's a link to the application: Coursera Static Web Page Link

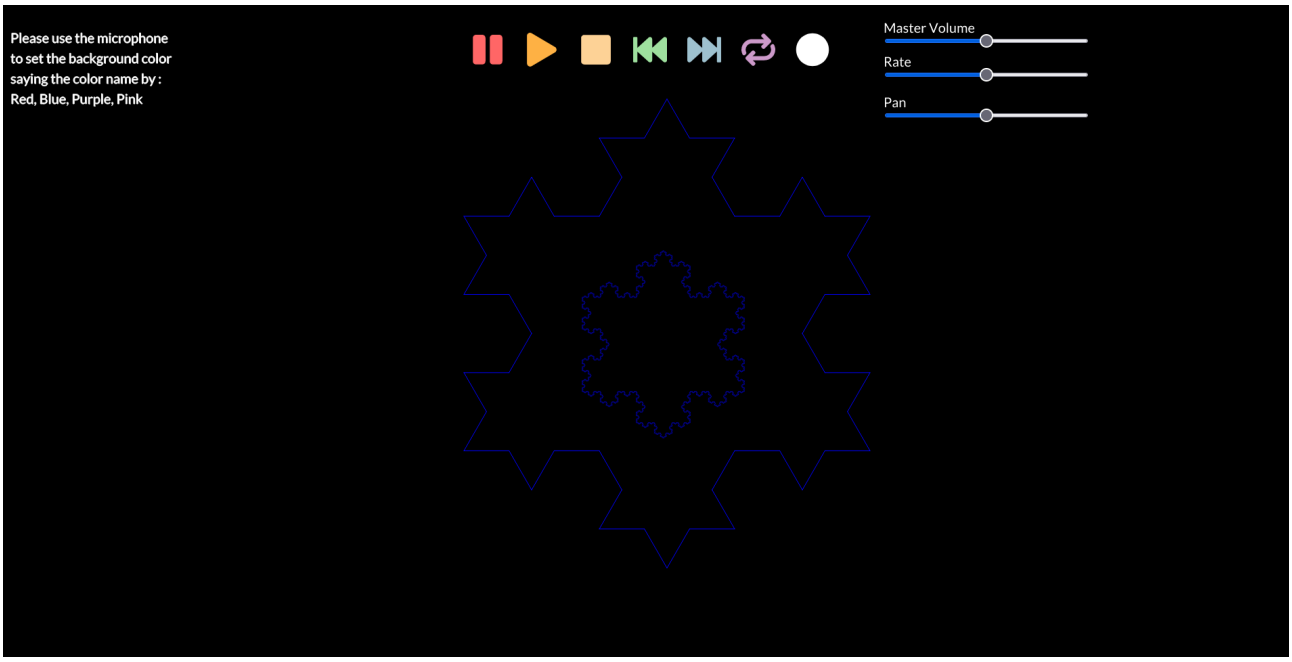

Figure 1: Audio Visualizer

2.2 Features

As you can see in Figure 1, the following features were selected from the signal to create an audio visualizer:

1. Zero Crossing Rate (ZCR): The shape of the geometric snowflake shape changes according to the ZCR of the signal. To do this, I applied to the Meyda library to extract the ZCR value and depending on the range of the signal's ZCR, I used a specific generation of the geometric snowflake to draw on the outer snowflake. I chose ZCR values because I noticed these values stay between 0 and 15 values mostly and used that as a condition to set the shape of the snowflake.

2. RMS: The snowflake is a collection of lines drawn in a specific way. The snowflake's line's thickness changes according to the signal's RMS values scaled by 10000 since RMS values are originally very small. I chose RMS values here because generally, that is a good metric to evaluate a signal on.

3. Energy: The inner snowflake vibrates slightly according to the signal's energy using the rotate function in p5.js. Since the energy values are large, the energy value divided by 100 was used in this case. I used energy as a feature here because I noticed there are moments when the signal's amplitude increases rapidly and wanted to find a way to visualize that aspect of the signal.

There are many other features included in this web application that makes it more functional. This will be further discussed in the next section.

2.3 Further Development

I included many different features in this application, also inspired by the audio player project. Here's a list of further development features implemented in this project:

1. Geometric Snowflake Implementation: Choosing to use a more complicated shape than a rectangle or a circle :
2. Speech Recognition: It was used set the background color of the web application using p5.speech. Due to library limitations, it only works on Chrome and does not work on Firefox. It first asks for the user's permission to use the microphone and then uses the user's input in English as the command.
3. Random Colors: The colors of the lines of the snowflake are chosen randomly
4. Pause Button: To pause the audio signal
5. Play Button: To play the audio signal
6. Stop Button: To stop playing the audio signal
7. Skip to the Start Button: To skip to the start of the signal
8. Skip to the End Button: To skip to the end of the signal
9. Loop Button: To turn the loop on or off for the signal
10. Record Button: To record your own audio
11. Volume Slider: To increase or lower the volume of the signal
12. Pan Slider: To increase or lower the pan of the signal
13. Rate Slider: To increase or lower the rate at which the signal is playing