Spotify AVis

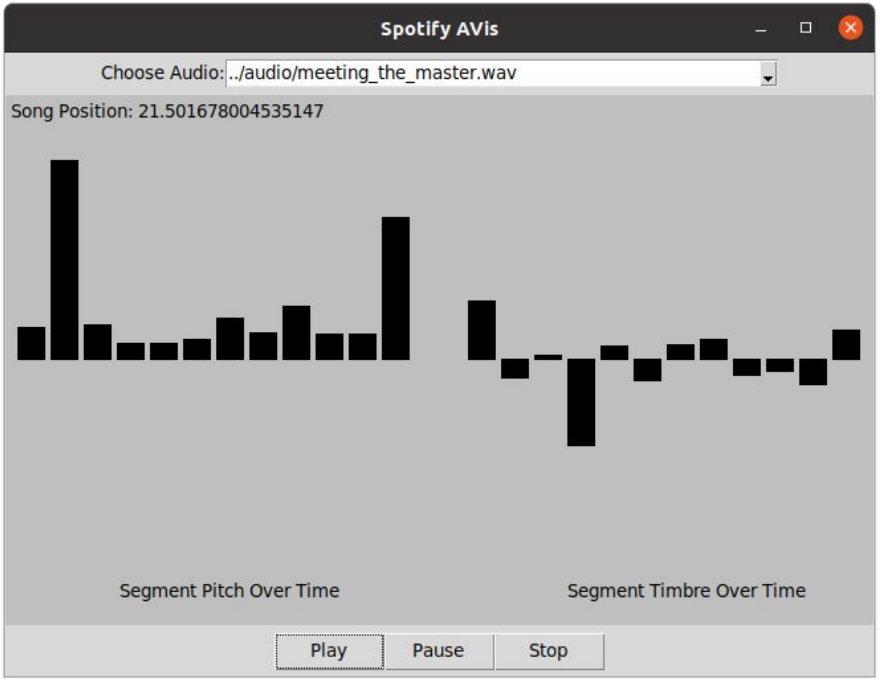
An audio visualizer utilizing Spotify audio analysis

Curt Bridgers

Graphical User Interface (GUI)

- Library used: TKinter

- Motivation: TKinter provided all elements I deemed necessary for the tool to function properly



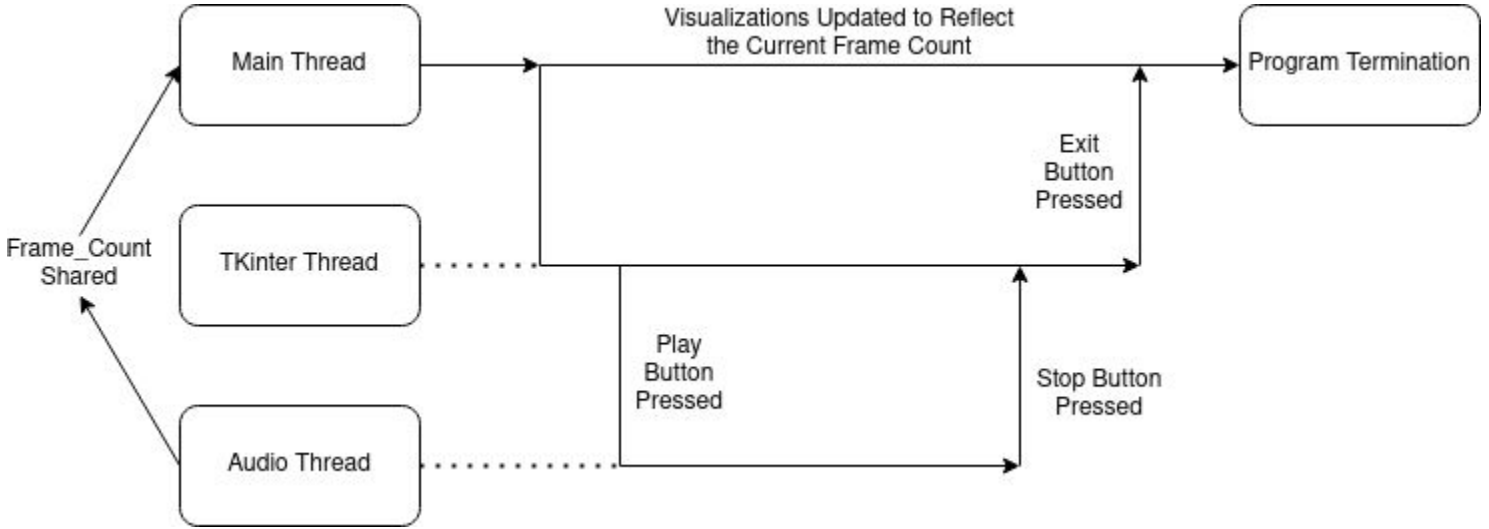
Software Architecture

Three threads:

- The main thread
- The TKinter main loop thread
- The audio player thread

Shared State:

- Current frame count from audio to the main thread



Playing Audio

Library Used: PyAudio and Wave

Challenges:

- Anaconda's package does not work (must use a python virtual environment or system python)
- To my knowledge, Librosa doesn't provide a way to read *n* frames from a wav file to use with PyAudio's callback function
- *.wav* is the only supported audio format

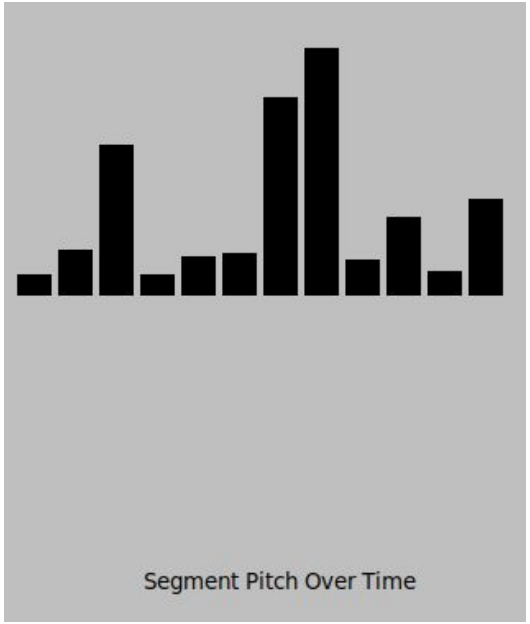
Spotify Data

Features of Note:

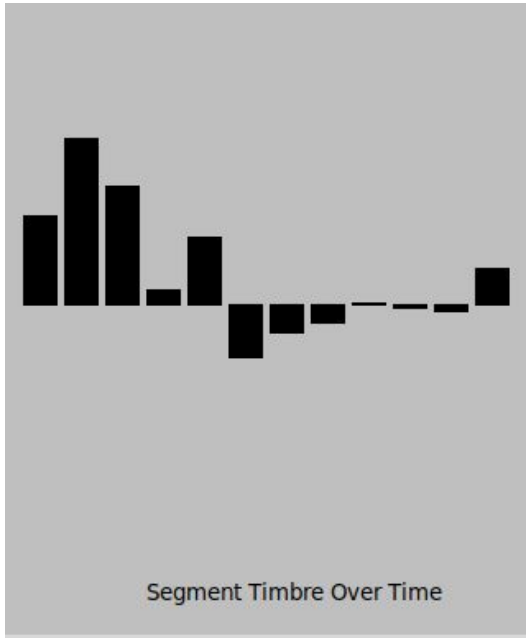
- Pitch vectors per segment -> (Frequency)
- Timbre vectors per segment -> (Tonal Quality/Texture)


```
{
  "start": 2.54544,
  "duration": 0.38503,
  "confidence": 0.023,
  "loudness_start": -47.005,
  "loudness_max_time": 0.21842,
  "loudness_max": -43.746,
  "loudness_end": 0,
  "pitches": [0.092, 1, 0.092, 0.015, 0.008, 0.047, 0.018, 0.018, 0.431, 0.022, 0.017, 0.006],
  "timbre": [14.443, -110.919, -91.914, -88.489, 84.812, -57.273, 14.35, -8.498, -6.895, 7.28, -3.634, 9.016]
},
```

Visualizations



- Pitch and timbre vectors visualized as bars
- Pitch values scaled up
- Timbre values did not need to be modified in any way



Future Work

- Add expected functionality to the Pause button (currently it functions identically to the Stop button).
- Eliminate the necessity of obtaining Spotify's audio analysis externally.
- Add functionality to correct any discrepancy between the sample rate of the *.wav* file provided and its corresponding audio analysis.
- I would also like to add the ability to handle multiple audio formats as opposed to only *.wav* audio files.