TunerFish Final Report

Structure

Introduction

The main feature, the tuner will have 3 key functions in order to work. First, it is able to get audio information from a microphone into a buffer. From there, it forms the data in the buffer into usable wave-function data. With the wave-function data, a Fast Fourier Transform can be performed on the data to create a new wave function. This new function should contain peaks at frequencies which make up the original wave function. From this data, we look for the loudest frequency. This frequency should be the fundamental note of whatever it hears, and will be graded against. From this fundamental note, the app finds it's pitch. Once it knows the pitch, it compares it to a table of the standard tone frequencies in A=440hz. The program finds the closest pitch and compares it to what the program could hear. It then tells the user how many cents off as well as gives a visual indicator for how sharp or flat the signal is. 50 cents flat being all the way to the left and 50 cents sharp on the right.

The metronome feature is the most straightforward feature. It's a bunch of hard-coded checkboxes with a listed beat-per-minute and, when started, plays a sound at that timing. Sounds are done through the built in system beeps. The timing will be calculated by converting the bpm into beats per millisecond. Using that, timing, play the sound at that interval when the user says to start and stop when they say stop.

The Note Player will play F, G, A, B, C, D, E, F#, G#, A#, C#, D# notes from octave 1 to 6. If the user choose the sustained button, the note will be playing repeatedly until the button stop is pressed

The History feature will display History of Audio Analysis events. The events information will include date and time a note was detected, what note it was and how many cents off the sound was. If needed, the user can chose to clear all data or delete particular events

Technology

Microsoft visual studio is required to build this project. Using Window's Winform system, and C# libraries, the project can be compiled. Json is used for storing history data. Accord is used in order to take in microphone data. Math.net is used for the fast fourier transform

Future Work

The tuner would look more along the lines of the note-player for more aesthetic pleasure.

In future versions there will be an instrument oriented version of the tuner. This would include displays of strings, including a customizable amount of strings and tuning configurations for those strings. The strings would also start playing a note at their respective frequencies when it hears the user playing a note close to that string. The tuner will also be able to have a finer degree of accuracy, allowing more precise tuning. The option to select which microphone will be added. The option to choose or type the temperament

The selection form will be on all of the functions, not just a separate bar serving as a hub.

Known Bugs

Tuner:

- If the user manages to get the detected note high enough, the equation to overflow does not work and just crashes instead
- Does not release memory when exiting the function

Audio Analysis:

- Graph changes size rapidly from background noise, and from the y scale changing rapidly.
- Does not release memory when exiting function

Note Player:

- If the user try to play the note C#5, it crashes because the sound file does not exist
- If the user tries to play the note A4, it crashes because the sound file does not exist
- The order of the notes does not follow pitch. E is a lower pitch than B for example

History:

- Every time the tuner is started, the history clears itself

- If the user double click on the headers for the table, it gives the user the option to clear entry -1. Doing so causes the application to crash with an outOfRange exception

Metronome:

- The metronome does not stop playing even when closed.
- The metronome can have multiple checkboxes ticked
- The metronome doesn’t beep at the beats-per-minute that it says it does

Design

Classes/Scripts

Event:

The event class has the properties Date which is a DateTime, Note which is a string, and CentOff which is an integer.

Note:

The note class is inherited by many of the other classes. It includes important properties that are useful when interpreting musical notes. Such properties include the note name, frequency, octave, and for the program, an index to where that note is on the pitch table. The only method is the constructor.

Tuner:

This class is used for the TunerForm and the AudioAnalysisForm to perform FFT duties as well as Note handling. A noteFrequencyTable, which is an array of Notes, is a list of all the notes with their respective frequencies and octaves. The Tuner class handles the FFT duties, only needing an array of doubles representing the microphone input, and will return an array of doubles representing the Forward Fast Fourier transform. With this data, we can find the loudest note, and assume that is the fundamental note. The Tuner class has a method findClosest which takes a frequency represented by a double and find the closest match to that frequency in the noteFrequencyTable, and return that Note. setupNoteFrequencyTable populates the array with the notes at runtime, allowing for further customization when temperament is added. The array noteNames is an array of all the names of the unique notes within an octave and is used for populating the noteFrequencyTable.

TunerForm:

UpdateAudioGraph updates the internal arrays used for processing the FFTs and the data associated with it. This includes things like cent calculations, differences between frequencies, finding frequencies, etc. UpdateCents is the cent specific part of UpdateAudioGraph. This calculates the Cent related factors, and updates the textboxes accordingly. FindNeighborNote calculates the closest neighboring note to the current found note. For example, an A4 which is sharp at 444 hz's neighbor would be A#4. RATE is the sample rate of the microphone, and dictates the sample rate of the buffer. BUFFERSIZE is the size of the buffer holding the microphone data. This buffer needs to be a power of 2 in order for the FFT to work. History is a list of events to be sent to the History function when exited.

AudioAnalysisForm:

UpdateAudioGraph updates the internal arrays used for processing the FFTs and the data associated with it. This is mostly just finding the loudest note, and finding the pitch and FFT index of that note. RATE is the sample rate of the microphone, and dictates the sample rate of the buffer. BUFFERSIZE is the size of the buffer holding the microphone data. This buffer needs to be a power of 2 in order for the FFT to work. bwp and the waveIn variables are used to store and get the microphone data into the program in the form of byte arrays.

SelectionForm:

A simple user interface consisting solely of buttons. Each button creates a new form for each of the functions. It passes them a reference to the selectionForm, shows the new form, and hides itself. Then it waits until the called form closes and makes the selectionForm visible again.

Database Schema + Files

Data for History feature is stored in a json file. The data is first gotten from Audio Analysis then stored in an Event Object. Said object is stored in a List of Events. As soon as the user stops the Audio Analysis feature, said List of Events is serialized into the json file.

To get the data into History Feature, we simply deserialize those data into a List of Events.

Non-Code Files

.wav files that contain notes sound
Json file that keep all the data for History Feature

Links

Diagram file: TunerFish-Design.drawio - diagrams.net

User Interface

When the application begins, there will be five buttons. Each button is clearly labeled with the functions. When the user clicks the button, the selection buttons disappear, and the function shows itself. When the user wants to switch function, the user closes the window of the current function, and the selection buttons show back up again.

Installation

List of Tools

- Visual Studio
- C# Default Libraries
- Accord
- Math.Net
- Json

How to Deploy

Install Visual Studio 2017 or 2019. Then select the project solution. After opening the solution, press the green button.