##############################################################################

Text, logo

Description automatically generated

README.txt

Emilio Santana-Ferro, Rasheed Martin, Mram Shalabi

##############################################################################

Introduction

This code is created with the intention of combining multiple waves into audible frequencies which is done by creating tones in the range of 300 Hz ≤ f ≤ 3 KHz, which is the range for the human ear (give or take). 4 distinct “notes” are defined along with their standardized frequencies and are placed into vectors (c, d, e, ed), which then when reproduced at certain times will create a “song” for us, in this case Mary had a Little Lab.

Furthermore, the code can also generate plots and a spectrograph which will visually detail all the aspects regarding each individual frequency and can help us better understand and analyze what is occurring.

* Goals and Capabilities:
  + Create and analyze Time and Frequency graphs representing the audio track
  + Create a Spectrogram of the frequencies being used in the track
  + Add white noise to audio track and then be able to remove it with the assistance of a filter built into the code
  + Re-plot and analyzation of the Time and Frequency graphs for the cleaned-up signal

---------------------------------------------------------------------------------------------------------------------

Getting Started:

* Prerequisites:
  + PC or Mac running Windows 10 or Mac OS El Capitan 10.11.06
  + MATLAB license (Provided by NYIT)
  + MATLAB downloaded and Installed
* Installation:
  + Check OS version and update if necessary
  + Download the installer at <https://www.mathworks.com>A picture containing graphical user interface

    Description automatically generated
  + Sign up using your NYIT email to receive free yearly license
  + Pick your architecture (Most current PCs are x64-bit)
  + Run through the installation

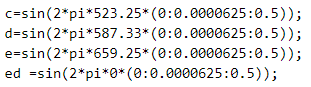
\*\*\*BE MINDFUL OF YOUR OUTPUT SAVE FOLDER, THIS IS WHERE YOUR .WAV FILE AND .FIG FILES WILL BE PLACED AFTER CODE EXECUTION) \*\*\*\*

* + Login with Mathworks account to activate license
  + **In order to perform the operations below you will need to download the Signal and Process ToolBox and the Communications ToolBox as add-ons!!!**

------------------------------------------------------------------------------

Uses:

This Section will relate to functions used in the project and how they are being used to get a 10 second audio wave file.

* Functions:
  + **Time Vector**
    - For the time vector, the **linspace function** was to create a vector from 0 to 10.013 seconds with an evenly spaced numbers of 80010
  + **Note Declaration**:
    - This is defining each “note” with a vector of sine waves using the **sine function** to create this vector. Each vector was created with its own frequency. The vector of ed is an arbitrary vector made to create a pause in the music.
  + **Saving AudioFile**
    - This saves the audio files using the **audiowrite function** which takes in the parameters of the song title, the input signal, and the sampling frequency.
  + **Reading an AudioFile**
    - This reads the audio file using the **audioread function**  which takes the parameters of the song title and returns the input signal and the sampling frequency
  + **Plotting the Signal**
    - This plots the signal using the **plot function** which takes the parameters of the time vector and the signal.
  + **Plotting the Signal on a Subplot**
    - This will plot the signal on a subplot by using the **subplot function** which takes the parameters of the grid (ex 2 x 2) and where on the grid you want to place the plot.
  + **Saving a Figure** 
    - This will save the figure to the file directory by using the **savefig function** which takes the parameters of the title of the fig file you are saving it as.
  + **Absolute Value of Fourier Transform of a Input Signa**l
    - This takes the absolute value of the fourier transform of the input signal by utilizing the **abs function**  and the  **fast fourier transform function**. The **fast fourier transform function** takes the parameters of the input signal and the number of the samples. The **abs** **function** takes in the parameters of an input array.
  + **Plotting a Plot on a Specific Figure**
    - This involves declaring and initializing a figure object
    - Then using the **figure function** you can assign a plot to the specific figure you want to plot it on.
  + **Creating a Spectrogram**
    - In order to create a spectrogram for the input signal, the **spectrogram function** has to be utilized. This takes in the parameters of an input signal, a window, and other parameters. Type the spectrogram function into the help box in order to see the documentation. In this program, the **hanning function**  was used with a length of 2000 as that gave the best frequency and time resolution for this project. If that does not work for you, try to play around with the number and tune it to your liking.



* + **Adding White Gaussian Noise**
    - In order to add white gaussian noise to an input signal, the **Add White Gaussian Noise function** is used which takes the parameters of an input signal, a signal to noise ratio (db), and signal power.
  + **Creating a Bandpass Filter**
    - In order to create a bandpass filter, the **bandpass function** is used which takes in the parameters of the input signal that you want to create a filter for, the band of frequencies you want to filter, and the sampling frequency.
  + **Playing a Sound from MATLAB**
    - In order to play a sound from MATLAB, the **sound function** is used which takes the parameters of an input signal and the sampling frequency. 