Project Proposal:

Project Description: My project, entitled “PyGarageBand” (for now), is essentially an audio manipulating program made using python. Essentially, my program seeks to emulate GarageBand by letting the user create their own music masterpieces by importing .wav files, modifying them, creating them, and combining them all together. The modules I are using to complete this project are PyDub, TkInter, and Time

Competitive Analysis:

Evidently, my program seeks to recreate GarageBand, an already existing platform that allows users to create their own music. Since my TP is based on this program, I’ve tried to add as many features as the normal GarageBand as possible: users can create melodies on a virtual keyboard, can input pre-made sounds such as drumbeats and keyboard melodies, as well as have a platform where they can manipulate and combine the audio files they import. However, my program will lack certain aspects of GarageBand, as my program will not have a beat detection feature or built-in tune detection.

Another program that already exists that is similar to mine is called pyBand, and was made by a 112 student in S19. This program had the same basic premise as me, which was to create a program that enabled users to make their own music. My program, like pyBand, will enable users to input files, manipulate them (crop them, overlay them), but will have the added feature of making music within the program itself. My virtual piano enables users to make melodies that they are then able to layer on top of existing files they have. Also, my program will have 2 canvases with user information, as opposed to the one that pyBand has. The first canvas will be where the user can input audio files, simply overlay them, and make tune using the piano. Then, on the second screen, the user will be able to manipulate the files they imported to make their final masterpiece.

Structural Plan

My project will be organized as two main canvas screens that will be made using 2 separate classes for each different screen. Each class will have its own set of functions, including the __init__ function, initializing variables, as well as the helper functions which will outline various things that the screen will have (including drawingSongs, playingSongs, converting notes to frequency pitches that PyDub can then generate sound for). The first class will revolve around gathering sounds, and the second class will have to do with maniplating it to get the finer details of the music sorted out. So, my program will revolve around OOP and will have a class for every screen and a function to toggle between the different classes. The variable that will be shared between both classes is the self.fileNames. I have also added another class to the beginning of my program, which controls which class is running at a particular time.

Algorithmic Plan

The hardest part of my project will be the second class with the second screen, which has an interface where users can drag (or select) the song they want to use, manipulate it by cropping it, splitting it, and combining it with other audios. I will adopt the style that the student used in pyBand, in which audio selected will be able to be placed in a stream and then manipulated. In order to do this, I will separate each audio into chunks (TkInter buttons) based off the length of the audio segment, and then each chunk can be manipulated, as well as the audio as a whole.

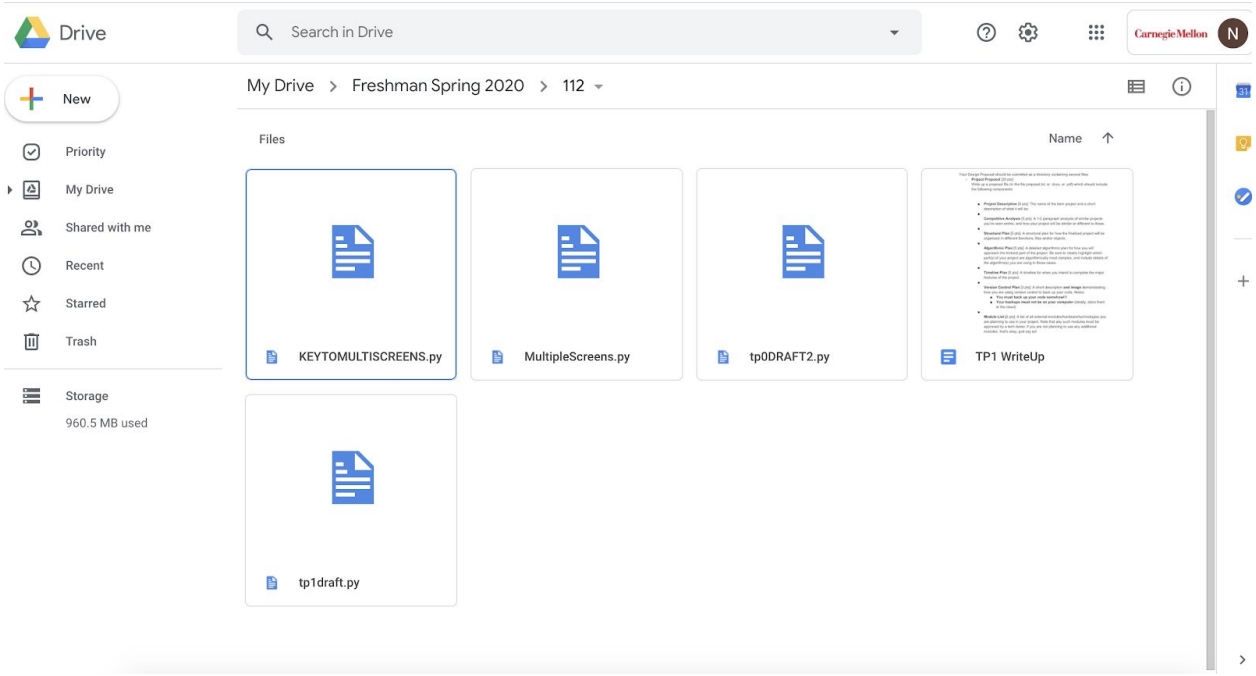
Another algorithm that was difficult to make was the algorithm that converted a combination of keys pressed on the virtual keyboard to an audio file that would be placed in the user's directory and accessed in PyGarageband. In order to do this, I first made a helper function that needed the note letter, octave, and duration in milliseconds. Then, based off of the octave, a list was created with the frequencies for each note(C, C#, D, D#, etc), Then, depending on the note, the particular input of the list was sought. Then, a function in PyDub converts the frequency in the list into an audiosegment. After this function returned the audio segment, it was added to a result audio segment. Then at the end, the result was exported and then added to the list of songs to draw. (Basic description of a more complex process)

Timeline Plan

4/16/20: TP 1 meeting -- Understand how to toggle between screens, begin working on the second screen.
4/18/20: Get the base of the second screen class (algorithm set) as well as start adding the built-in instrumental sounds
4/20/20: Add finishing touches such as decorative items and add features that enables users to enhance the video further
TP 2 Checkpoint: Finish most of the second screen class -- reach MVP

Version Control
My program will be backed up based on each TP checkpoint. All the deliverables will be saved on my directory, but will also be saved on google drive, so it can be easily accessed.

Here is the proof that it is on Google Drive:



Modules

- PyDub
- PyAudio
- TkInter
- Time

Storyboard Attached in Zip File