

# Project Proposal

- **Project Description**

- Visual Harmonies

- This program will take in a piece of music as a .wav file, analyze the file, and then create a moving pattern of shapes and fractals that reflect the data from the music analysis.

- Competitive Analysis

- There are many programs, such as RenderForest and Spectrum that create a visual representation of music, but they focus mainly on drawing the wave plots of the audio while it is being played. They have features such as changing the colors (though randomly) and changing the amplitudes of the waves. My project will be similar to these projects because I do plan on taking audio data and creating a visual representation of the qualities in the music. However, my project is different because I incorporated many different shapes to complement my visuals, such as recursive fractals and rotating shapes. Additionally, the colors in my music will reflect the changes in frequency within my music, giving purpose to them instead of just having them randomly change colors.

- Structural Plan

- I will have three different sections to my code. The first section deals with audio and recording, the second section deals with the visuals of my code, and the third section deals with putting the other two sides together.
      - The audio section of my code both records, plays, and analyzes the music. In other words, there is a function that records and save files, there is another function that takes in a wav file and plays it, and there is one more function that processes a wav file and returns an array of amplitudes recorded while analyzing the file.
      - The visual section allows a number of vertices to be inputted and creates both shapes and recursive fractals that change based on the number inputted. The shapes start out small in the middle of the canvas, but rotate and grow larger with each rotation. The recursive fractals disappear whenever there is a new shape that grows from the center of the screen.
      - The third portion of my code takes in the musical analysis data and allocates certain amplitudes and frequencies with different colors and number of vertices. Additionally, it uses time to sync up the patterns while the music is playing.

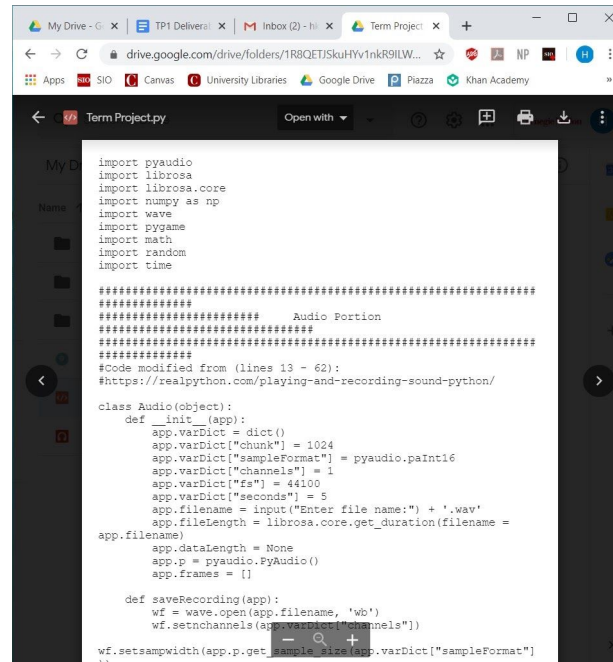
- Algorithmic Plan

- One of the most challenging aspects of my project was coding recursive fractals. My approach to this part of my project was to create a class (Fractal) that would keep track of the number of vertices that is inputted into the function. Then there are functions that get the endpoints of the fractal. After it gets the endpoints, then it draws the lines between a constant point and the list of other points. Afterwards, it recursively goes to each of the individual points in the old endpoint list and creates more lines and new endpoints from each of those endpoints. I had to code to strategically account for the different angles and make sure that the lines end up evenly spread out around the point. With each recursive loop, the length of the lines decreases, and if the line length reaches a certain length, then the recursive function stops calling itself.
- Another challenging aspect of my project is translating the audio information with the visual functions and syncing up both the audio files and the visuals. I believe that I will use the amplitude data and if the data point has a large value, then the RGB values on the shapes and fractals would change to become brighter, (increasing the values for two colors). If the amplitude is smaller, then the RGB values will decrease to leave the color more desaturated and less vibrant. Additionally, if the frequency is higher as well, then the recursive fractals will have more points, drawing more on the screen. In order to sync up the music with the audio, I plan to use two arrays/lists, one to keep track of the frequencies/amplitudes, and the other to keep track of the time. I would check what the time (in milliseconds) into my list, and then use the index to find the value in the array with the amplitudes/frequencies. That way, my music can be aligned with my visuals.

- Timeline Plan

- April 16th: Turn in this deliverable and have my term project analyze the amplitude data and draw fractals and shapes based off of the amplitude data (no matching up to the music) by the end of the day
- April 18th: By this point, I hope to have the frequencies detection also figured out. I also hope to have a user interface figured out by then as well.
- April 20th: I hope to achieve MVP by this point. I will have software that reliably intakes music, finds its frequency and amplitude and changes
- April 22nd: At this point, I'm exploring more options to sync up my music and visuals. I also will be looking at more recursive fractals. Thinking of different ways to increase the complexity of my project.

- April 29th: Turn in term project
- Version Control Plan
  - I plan on uploading my Term Project Folder into my Google Drive in order to have it on the internet so that if my computer crashes, I have it on my drive. I plan to do this every time I complete a milestone in my project.



```

import pyaudio
import librosa
import librosa.core
import numpy as np
import wave
import pygame
import math
import random
import time

#####
##### Audio Portion
#####
#####
#Code modified from (lines 13 - 62):
#https://realpython.com/playing-and-recording-sound-python/

class Audio(object):
    def __init__(app):
        app.varDict = dict()
        app.varDict["chunk"] = 1024
        app.varDict["sampleFormat"] = pyaudio.paInt16
        app.varDict["channels"] = 1
        app.varDict["fs"] = 44100
        app.varDict["seconds"] = 5
        app.filename = input("Enter file name:") + '.wav'
        app.fileLength = librosa.core.get_duration(filename =
app.filename)
        app.dataLength = None
        app.p = pyaudio.PyAudio()
        app.frames = []

    def saveRecording(app):
        wf = wave.open(app.filename, 'wb')
        wf.setnchannels(app.varDict["channels"])
        wf.setsampwidth(app.p.get_sample_size(app.varDict["sampleFormat"])
        )
  
```

- Module List
  - Pygame
  - PyAudio
  - Librosa
  - NumPy
- TP2 Update
  - Because of the nuanced nature of frequencies and especially with many possibly different melodic lines, it's nearly impossible to track the frequencies of a melody. Also, I was not able to find a librosa feature that could track the frequencies accurately. The frequencies would have affected the number of vertices my shapes had while they were moving on the screen, and in order to replace this, I had the tempo change that feature instead: the faster the tempo, the more vertices.
- TP3 Update
  - Firstly, I added one more pattern in my program. It is implemented by iterative code that draws many circles in a set pattern.

- Secondly, I figured out how to do frequency part of the project. The STFT returns a matrix in which the columns represent time increments and the row represent frequency bins. By tracking the bin with the greatest frequency magnitude at a given point in time, I was able to pin the bin whose frequency was greatest. By taking that bin, I related it to the number of vertices the shape would have.
- Because I added the frequency, the structure of my code also changed. After calibration, my code creates an events list that shows which patterns will be displayed and in which order. I had to carefully take into consideration the timing of all the different patterns and the colors (since each pattern takes in a different number of rgb values). So for each chunk of time while a pattern was running, I took the appropriate number of rgb values and displayed them onto the screen. The vertices were also determined while the event list was being created because the number of vertices a shape had also influenced the time it was on the screen. In order to do this, I had to create a list that recorded the frequencies every second or so in order to figure out the average frequencies as well as the lowest and highest frequencies. This list's sole purpose was to figure out this information so that the program can get the actual frequencies and vertices when the patterns were running with the song. So after that, I created a function that would return the number of vertices for a given frequency. I made sure to relate the time and the color values as well as the patterns to make sure that the visuals and the music were all synced up.