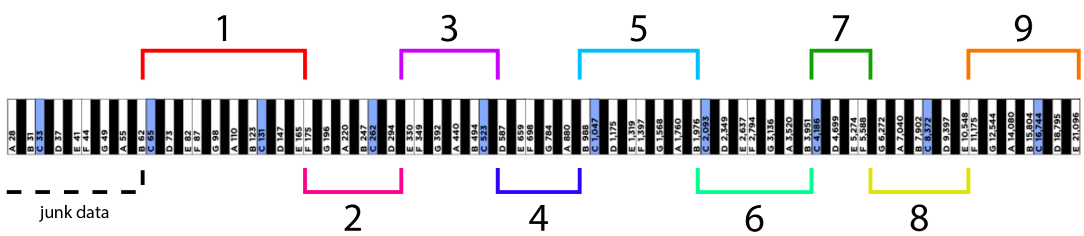
Adam Schmieder
6 December 2021
MAT 240A

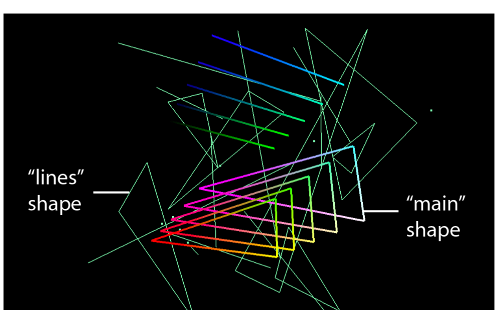
Final Project Report: Piano FFT, Visual Response Instrument

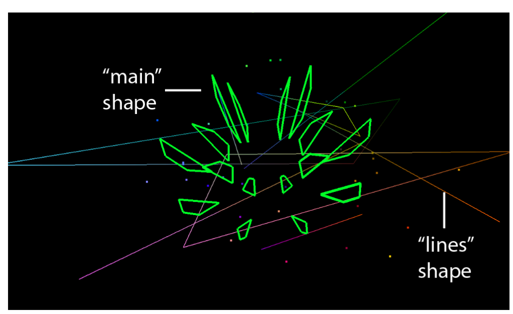
The goal for this project was a "visual instrument," that would analyze the frequency data of a given audio signal, and utilize that data in various ways to generate complex visuals in live time, using Max MSP and Jitter. In particular, this instrument would focus on the frequency data of the piano, and would be crafted to respond best to the piano's audio data and its qualities (frequency amount, note onset, bass tones, speed of rhythm, etc.). This project would be less of a direct mapping or conversion of the audio data into its visual response, but more a careful and intentional assignment of this data into certain aspects of visual generation, that I have deemed adequate; where I have much of the creative sway of how things will look, in response to a set of audio data.

Using code written by Karl Yerkes, which took the Fast Fourier Transform of a given signal, and converted that data into a list of frequencies (each with their own respective amounts present in the signal, scaled from 0. to 1.), I was able to curate frequency data into certain aspects of visual generation which I deemed as interesting. I split up the list of frequency data into 9 bands of information, which were carefully chosen to accurately reflect certain timbral and registral qualities of the piano:



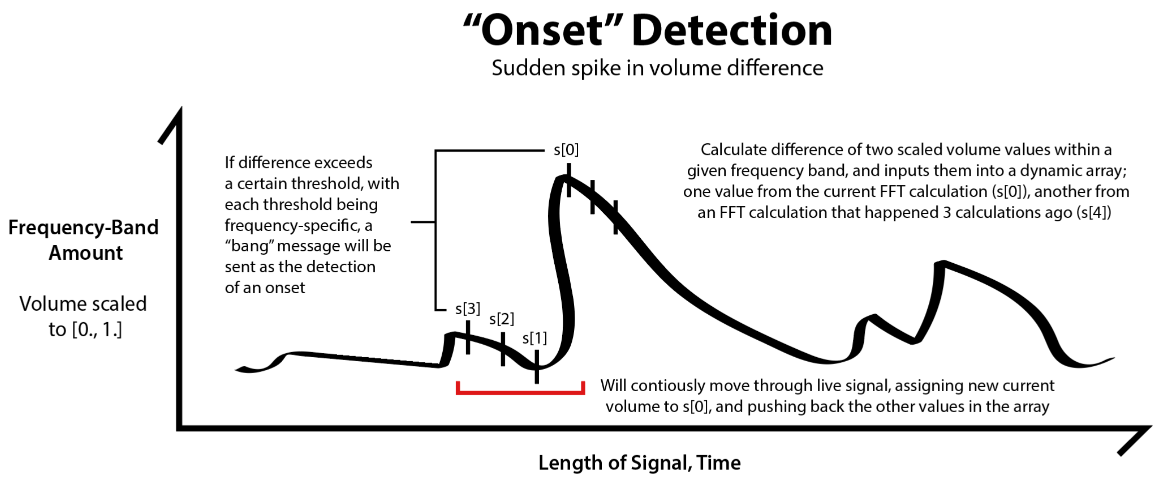
And used this data, in addition to further calculations of this data, in order to produce transformations within two visual objects; Open GL primitive shapes using jit.gl.gridshape (labeled as "main," present in the foreground), and a matrix of sprawling lines and points using jit.gen and sin/cos transformations (labeled as "lines," present in the background).

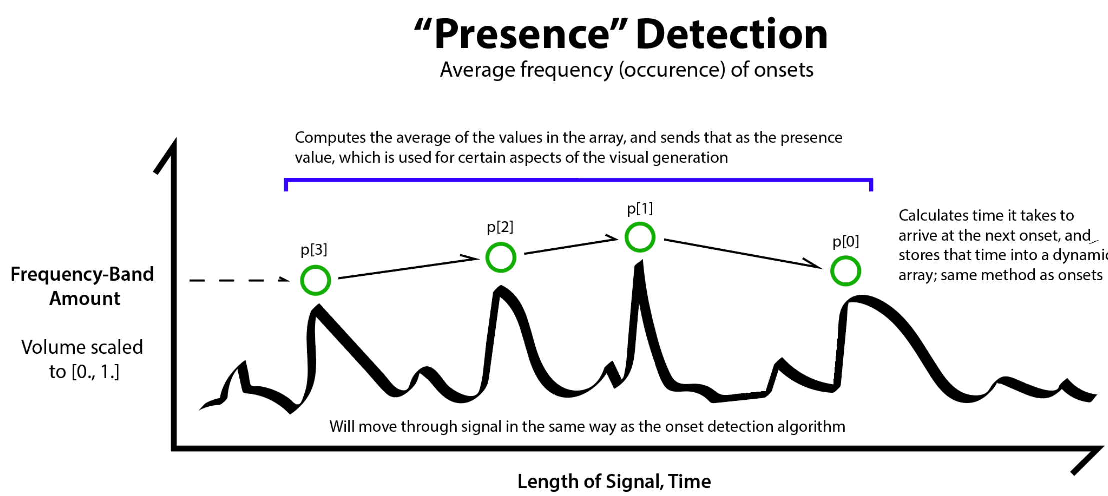




In order to understand *how* to generate these visuals, I studied the code found in two programs; "Ribbon": https://maxforlive.com/library/device/4041/ribbon for the "lines" generation, and "Geometrum" https://www.sabinacovarrubias.com/geometrum for the "main" shape generation. Understanding how to actually generate visuals like this was the most laborious of this project's endeavors, as I came into it with little to no understanding of this process, and thus required lots of Open GL tutorials and code dissections in order to grasp how to replicate this process.

Using the FFT data, I performed two additional calculations; onset detection, and “presence” detection (the frequency of onsets), which are explained in detail below:





These detection methods were the crux for how I decided visual generation. I’ve broken down each of the data methods I used, and the visual results that I decided they would give:

“Main” shape	“Lines” shape
Direct FFT Data: - Shape will turn according to volume amount; higher volume, faster turn, and vice versa **Onset Detection:** - Shape changes if onset in bands 6, 7, or 8 - Color changes if onset in band 1 - Shape will “bounce” if onset in bands 4, 5, 6 **Simultaneous Onset Detection** (>1 at same time) - Color mode changes if onset in bands 5, 6, 7, 8 - Draw mode changes if onset in bands 3, 4, 5, or 4, 5, 6, or 5, 6, 7	**Direct FFT Data:** - FFT list is fed into a jit.matrix, which is fed into jit.gen, where sin and cos transformations take place to give the lines their shape. Makes the lines jump and move in response to the audio **Onset Detection:** - Color changes if onset in band 2 - Color mode changes if onset in band 1 **Presence Detection:** - Changes draw mode according to presence of bands 5 and 6. More complex drawing modes are chosen for higher presences, and vice versa