Sandhya Iyer

Advanced Creative Coding Final Project - “Computed Synesthesia

12/04/2023

Inspiration and concept:

My concept for my final AET 350C Project was inspired by my love for art and music. I’ve always been very interested in the arts, and in high school I did a watercolor project based on the work of Wassily Kandinsky, who is a painter with synesthesia. Synesthesia is a psychological phenomenon that causes sensory crossovers, activating two or more senses when there’s only reason for one to activate. Many people have sound-color synesthesia, which causes you to see colors as you hear music. I wanted to draw inspiration from this phenomenon for my final project, and decided to program “computed synesthesia”, a generative art program that turns musical input into artistic visual output.

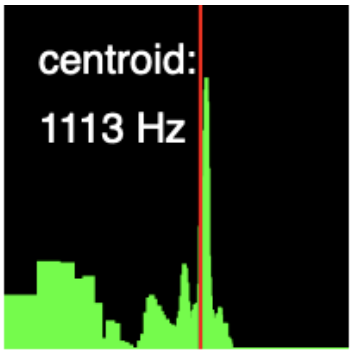
Development Process:

I started out with planning the basic structure of my code and planning the functionality. My fundamental goals were to have the mic take audio input, for that audio input to be interpreted in terms of amplitude and frequency, and then for a visual output to take place. I quickly realized that the “splashes of color” were aesthetic details that could be added later as stretch goals, so I started with generating circles on screen.

My first, most basic iteration involved a circle that changed size with the amplitude, or volume. This functionality was simple to implement, given the p5 sound library. However, recording the frequency input is what took up most of my time, and my approach to it changed

many times throughout my iteratior

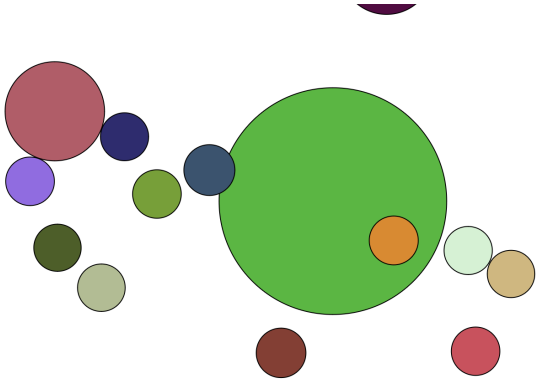
I started with attempting to change the transparency, or alpha value of the test circle through higher and lower pitch. Since the frequency is recorded as a spectrum, I wanted to find a way to get a single value from it. I read up on the p5 reference library and found the method getCentroid(), which got the Spectral Centroid of the frequency, which is the center of mass. I used Fast Fourier Transform and got the spectral centroid of the frequency, which worked, but was not as precise as I was hoping.



Key milestone 1: basic implementation of audio to visual controls

Once that was implemented, I created a random drawShape() function that generated these circles at a size and frequency relative to the amplitude and spectral centroid recorded from the mic. The circles were just being randomly generated every 2 seconds. However, at this point

I started to notice that the appearance of these circles on the page was happening in a very choppy, sudden way, rather than gradually animating in.



It was then that I remembered the Bloom project I had done for Introduction to Creative Coding last semester, which incorporates animation of circles growing onto the screen and then fading away. I referred to this code, and edited mine to have similar functionality. I moved the circles from a function to its own class, which gave me a lot more control over individual specifications.

Key milestone 2: circles become object, new animation for them

Next, I made the decision to generate a new circle only when a new note was detected, rather than just every few seconds. I felt that this would better represent the music played. To do this, I needed to assign certain frequencies to note names. First, I tried importing a csv spreadsheet linking frequency values to notes, and then I tried manually putting in the frequencies into an array and the notes into a separate array, and then matching the indexes to each other. However, the process was becoming complicated so I started to research some "tuning programs" that converted frequency from an audio input into note names. I found a Pitch Score project by Pierre Rossel that utilized a pitch detection model from an ml5 library to automatically convert the frequencies into notes, and I hadn't even thought about libraries as a

way to do this until now. I used this model in my code, and implemented some of the conversion coding from Rossel's Pitch Score. The text on screen displaying note name, midi value, and volume really helped me to start fine tuning the functionality of the project.

Key milestone 3: incorporating pitch detection through ml5

My final action involved adding buttons for interactivity. Although I had plans to make a color palette picker and other buttons, I had to boil it down to the essentials due to time constraints: a start mic button, a stop mic button, and an export button. These were all programmed as objects, and I simply moved code around to incorporate them.

Key milestone 4: adding buttons

The final project generates a new circle when a new note is over the volume threshold and holds for about a second. I ended up removing the amplitude aspect because I liked the aesthetics of the final art without it, but I would want to incorporate it somehow in the future or with more time. The pitch detection works to generate the color and placement of the circles, and the buttons let the user stop, start, and export their artwork. In addition, the text at the top provides information about the note being played, the midi value, and the volume.

Reflection:

Overall, I'm quite happy with the final outcome of this project. The functionality of it was the most fundamental goal for me, and I'm happy I got that working- especially the pitch detection and circle generation through a class and array. As for the aesthetics, I like the circles art, but I do believe there's room to expand and improve as well, especially in the user interface. I'd like to experiment with creating a growing canvas for longer songs, incorporating more buttons, adding a splash screen, and maybe playing around with different shapes on screen.

Note: B - midi 35 - volume 0.00

Examples of generated artwork: