Seunghyeon (Hyeon) Kim

Eric Autry

CSC-151-03: Functional Problem Solving with lab

December 13, 2022

Harp Machine

Overview

For the final project, me and my teammates created a harp machine. The purpose of the harp machine was to upgrade the beat machine from the third mini project utilizing the user interface (canvas) and making the decay rate of the sound more realistic. Not only the decay rate, but we also considered the linear combinations of different waveforms so that it can produce a more realistic sound wave (refer to formula 1). Our initial plan was to utilize this function to produce the note of piano. However, piano requires Fourier series to produce the note of higher and lower frequency. Thus, we thought "Why not make this into a harp beat machine." Harps have similar structure to the inner design of piano, so we decided to use the function to describe the sound wave made by harps.

Contribution

To approach this project, we decided to break down the function (see figure 1). The functions are broken down like the structure of a binary tree where each of the branches are the helper function of the head. In the break-down, I was supposed to program the harpNt branch and parNotes branch. As there were so many functions that we had to write as a group, we thought that writing the docstrings of the functions were required so that we could read each other's code when passed on.

While programming the harpNt branch, I ran into a lot of problems such as what function I should use to make the decay rate of sound more realistic, how I should modify my mini project 6 code to make the "make-sample" function take in more parameters (e.x.

maximum amplitude of each waveform, wave frequency of each waveform, etc.), how I can save Scamper from complex calculation, and how I should convert the MIDI note value into wave frequency of the harp.

For the decay function, we decided to use an exponential decay function (refer to line 19-22 of the code). The function indicates the nature human ears such that when the sound reaches $e^{\frac{-1}{5}}$ of the original intensity, it will be negligible. Thus, when *t* in the function reaches the duration of the note, the intensity of the sound will be the $e^{\frac{-1}{5}}$ of the original intensity. One of our helper functions calculate the waveform value of the function by multiplying the scaling function and the original intensity function (refer to line 105 of the code).

Out initial attempt to extend the definition of "make-sample" was to limit the number of waveforms into 3 and add them all together. While doing so, I thought that the function is not readable. Thus, I decided to add one more function called "spcfcVecCalcFunc" to change the function into a recursive call so that it can take in indefinitely many waveforms and amplitude values (refer to line 70-83 of the code). Even so, I ran into multiple other problems such as failing to call the recursive call as I set up the wrong function or I called the wrong list. Such problems were minor, and easily fixable (I took the test-case function to test each of the helper functions defined inside the top layer function), but I think if I planned the function with more details, I think such problems would not have rose.

To save Scamper, I decided to utilize the "vector-map" function. Originally, I was planning to utilize the envelop and use "vector-map" to multiply the two vectors, but I noticed that sometimes the envelop would not return the vector with same length as <waveform>-sample function. Thus, I decided to create another function that calculates the function value at specific time t (refer to line 26~84). Applying this function with "vector-map" would reduce the time complexity of O(n*log(n)) into O(log(n)) as it uses a single vector-map function and involves a single vector into account. Also, this approach can solve

the problem of unequal length of vector.

To convert the MIDI value into frequency, I utilized the fact that difference in the octaves is exponentially proportional to the frequency of the note. They can be expressed as the second formula (refer to formula 2). I decided to make this function with respect to the fourth octave C note frequency. While most of the program being successful, I noticed that changing the sample rate will change the duration of the note as the "sample-node" does not take in audio context as an input. Thus, we decided to fix the sample rate to 21000 samples per second (refer to line 171 of the code).

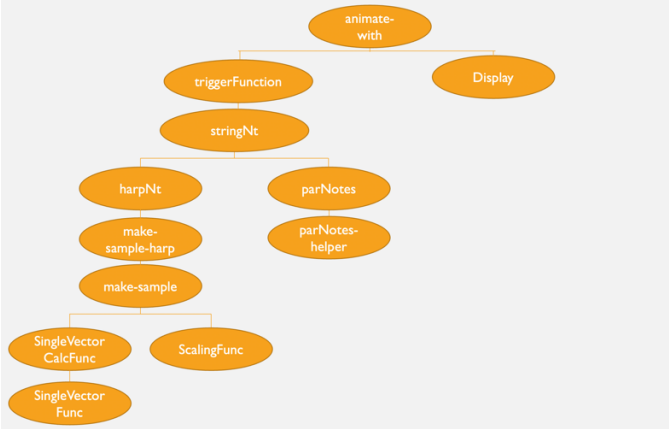
One of the problems we ran into but could not fix was the fact that there were functions that are equivalent to "play-composition" in music library are not present in the audio library. Due to this problem, we were not able to connect the harpNt function with the keys of the keyboard. However, we have implemented the keyboard with the MIDI notes like a piano keyboard. Surely it would have been better off with the implementation of "play-composition" in the audio library, but we chose to use the alternative which is still cool.

Conclusion

One of the worst problems I experienced while programming this project was that I did not turn on my auto save which I had to rewrite the whole code 2 times. Though a rookie mistake, it is one of the life lessons I should keep for the rest of my CS career. Though we could not fully implement the front-end part of the harp machine due to technical issues, I was able to apply the materials I learned from the CSC-151 class such as docstring, time complexity of the functions, "test-case," etc. through the final project. As a result, I was able to learn how to solve functional problems that I encountered while programming the code.

Figure

Figure 1



Kim, Seunghyeon. *Screenshot of Breakdown of the Code for the Final Project*. 13 Dec. 2022. Author's personal collection

Formula

Formula 1

$$T = time\ period\ \{same\ as\ reciprocal\ of\ the\ wavefrequency\}\ (s)$$

$$t = current\ time\ (s)$$

$$I(t) = -\frac{1}{4}\sin\frac{6\pi}{T}t + \frac{1}{4}\sin\frac{2\pi}{T}t + \frac{\sqrt{3}}{2}\cos\frac{2\pi}{T}t$$

Tasroc, Murey. "*Mathematical Equation for the Sound Wave That a Piano Makes*." *Signal Processing Stack Exchange*, 22 Jan. 2018, https://dsp.stackexchange.com/questions/46598/mathematical-equation-for-the-sound-wave-that-a-piano-makes. Accessed on 30 Nov. 2022

Formula 2

$$O = (MIDI\ value/12)$$

$$C_4 = \ frequency\ of\ C\ at\ the\ 4th\ Octave$$

$$f(O) = C_4 * 2^{(O-5)}$$

Subtracts 5 from O instead of 4 as 0th octave exists.