Concept Sketch – Pure Data Synthesizer

Concept

Initial Design

With the audio capabilities that visual programming languages provide over traditional programming languages, I plan on turning your normal computer into a synthesizer. By taking normal interfaces with home computers, Users will be able to use this synthesizer like traditional synthesizers or musical keyboards to create music. Users will be able to interact with this system with multiple ways to allow different ways of interacting.

First, the system will turn the traditional computer keyboard into the musical keyboard system that pianos and synthesizers use. Following Online Pianist system, each key will be assigned to a musical note, like a piano, then when a user presses this key, it will play the tone of that musical note. The keys will be arranged to model like the piano's interface to provide some familiarity of an established system. One row of computer keys, qwertyuiop, would be assigned to 10 white keys started on F or C, then the row above it, 1234567890, would be assigned to the black keys between each white key. For instance, if the key q is assigned the white keynote F, then key 2 would be the black keynote of F# and the key w would the white keynote of G. This pattern would continue for other rows of the keyboard until the arrangement of the keys no longer satisfy the model of a piano.

Another way for Users to interface with this system that I want to implement MIDI (Musical Instrument Digital Interface) support if users have the hardware to use. This technology allows users use actual musical keyboards to interface with the system like an actual synth instead of the nontraditional replacement interface. This interface can provide more features as MIDI keyboards come extra features, like knobs and sliders that can alter the sound. This interface will impact the system by having difference in loudness depending how hard the key is pressed or knobs that control pitch and modulation. Implementing this interface will provide for a smaller user group and is not required for the system but can be important for users that have the hardware and are already familiar of how it works.

Along with physical interfaces, there will be some on the program itself to further simulate the amount of controls that physical synths have. I want to focus on the core components of the Oscillator, Amplifier, Filter, Envelope, and Gates/Triggers. This will provide a basic synth and allow our users to create music. Along with some basic components, I also want to add in a vocoder into the system to allow more expression for my users. These interfaces will be done through Pure Data's GUI tools along with custom interfaces that are needed.

The data that the system will be processing would be audio signals. Synthesizers are audio manipulation tools, so the system must replicate that if it is to be a synth. Most of our manipulation of this data starts from the core oscillator as it produces the data, the audio signals, and along with the other tools that will manipulate this wave and the physical interfaces to change the oscillator to different signals or the vocoder combining two audio signals and resynthesizing them together from the user's microphone and the oscillator. The output of the system will be put through speakers for users to hear and an output device for users to record the output easily. This output device will output all the signals coming from the system to isolate the sound. Also, it can be used with the vocoder to use it in real-time so that users can have normal voice output through with no audio manipulation and then press a key to start the vocoder manipulation and code their voice.

Future Design

There are some future design ideas that can make the system even further developed and better. One idea is to change the fundamental algorithms of synthesis of these signals. For the vocoder, the algorithms use Fourier Transforms to break the signals and resyntheses them. There are other algorithms that can do this and, in my research, Ottosen and Dörfler create a vocoder with Gabor frames transforms to better slow down recordings. I would instead to use it to replace the Fourier Transforms used in the vocoder to combine the signals and see what it sounds like. It may sound the same or very different and I am intrigue on how this would affect it. The problem is that implementing this will not be easy. Pure Data provide objects, fft~ and rfft~, to do the Fourier Transform for us. I would have to implement Gabor Transform my own with the complex signal math. I am afraid that Pure Data can not do this easily and I am new to this kind math, so I am not making this a part of the current scope and would only work on it if I have time to fully understand and implement the Gabor Transform.

Technologies

I will be using Pure Data for the main programming language since it already has really good audio manipulation capabilities. I could use Max MSP as it is the closed-source version of Pure Data, but I want to make this program available to use by everyone and I don't think there is a way to compile in Max MSP to create a executable to use. Pure Data is open-source and can be easily (kind of) downloaded so I will be using Pure Data. I will probably use additional libraries like Cyclone to get some Max MSP objects to could make my project easier to program, or Gem to create some GUIs to better interface with the system. For hardware, as I have already written about, the system will require a keyboard to interface the system with, normal or MIDI. I have a normal keyboard as I would not be able to type this paper without it and I expect that most users interface a computer with a keyboard. The MIDI keyboard support will require additional hardware of a MIDI-compatible keyboard. I have already bought one for myself and they can be pretty inexpensive for the cheap ones. I will be using the one I already have to test the system with. More expensive MIDI keyboards have more features and interfaces to maniple the sound, but I will only support the controls that my MIDI keyboard will support.

Background Research

The research that I undertake was to find different ways of designing digital synthesizers. Two of my papers are about designing synths in a web application using traditional programming languages and libraries and the other is about designing a TUI for synths like the Reactable. Both papers provide some intriguing designs on how their users would interface the systems. I would like to implement some of their design into my, but for scope sake, I cannot. My system would focus on desktop application model and providing an interface that most people can interact with in the comfort of their own home. My last paper is about using Gabor transforms to time-stretch music signals better than Fourier Transforms. This

influenced me to change the core of the vocoder to try something different that could provide new unique sounds and more ways of expression. This will be for a future design for now, but it would be cool to paly around with this concept. My implementation would be focusing on mixing pitch instead of time-stretching. I have also provided some applications that influenced my design of the system. I have mention Online Pianist with it scheme on how it lays it keys on the computer keyboard and that I like this design. This application only focuses on replicating a piano, mine would be focused on synth applications. Midi.city is an online synthesizer that is closer to what I want to design. Midi.city provides a lot of libraries of different sounds, but it does not let you, or at least what I could find, customize these sounds like traditional synths. Mine application will provide more customization of the audio signals like a traditional synth.

Basic Timeline

First Week – 3/22 -> 3/29

Oscillator component and Keyboard Interface

Second Week – 3/29 -> 4/5

Amplifier and Filter components

Third Week – 4/5 -> 4/12

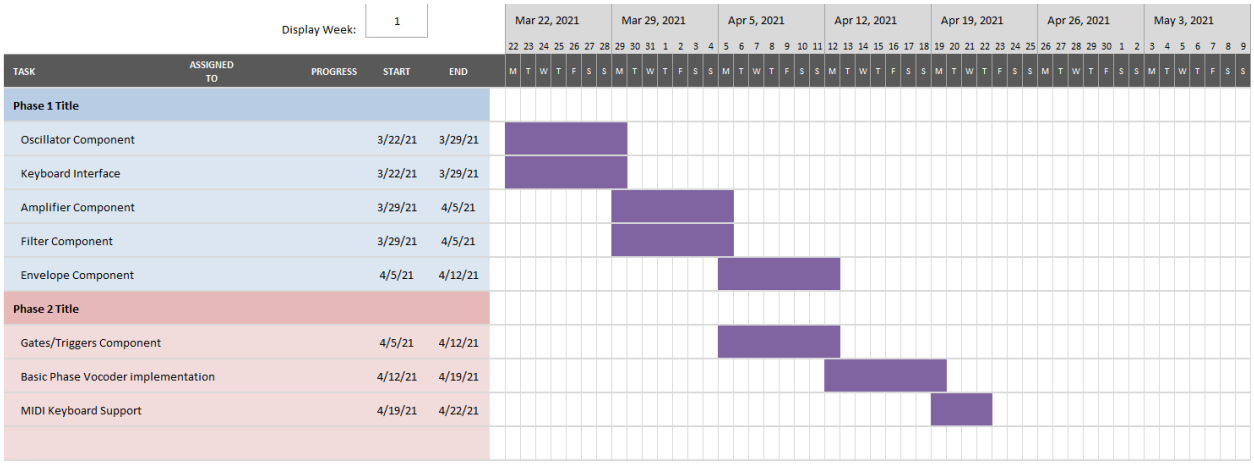
Envelope and Gates/Triggers components

Fourth Week – 4/12 -> 4/19

Basic Phase Vocoder implementation

Fifth Week – 4/19 -> 4/26

MIDI Keyboard Support



I am not too sure how long will the synth components will take so I will leave some buffer weeks if they take too long or implement other features if they take less then a week of development.

References

Edwards, N. (2019, April 15). Synthesizer basics explained - with audio Samples: Sweetwater. Retrieved March 22, 2021, from https://www.sweetwater.com/insync/synthesizer-basics-explained-with-audio-samples/

Ottosen, E. S., & Dörfler, M. (2017). A phase vocoder based on nonstationary Gabor frames. IEEE/ACM Transactions on Audio, Speech, and Language Processing, 25(11), 2199-2208.

Potidis, S., & Spyrou, T. (2014, June). Spyractable: a tangible user interface modular synthesizer. In International Conference on Human-Computer Interaction (pp. 600-611). Springer, Cham.

Roberts, C., Wakefield, G., & Wright, M. (2013, May). The Web Browser As Synthesizer And Interface. In NIME (pp. 313-318).

Midi.city: https://midi.city/

Online Pianist: https://www.onlinepianist.com/virtual-piano