Hardware neural net based synthesizer

John Tronolone (With Alex Hu)

1 Week of September 10

During the senior projects meeting the first week of school we were introduced to Cooper's Autoencoding Neural Network Synthesizer (CANNe Synth). This synthesizer uses a neural network to generate a single note of a sound generated by an autoencoding neural network, affording the creation of entirely new sound palettes. This coupled with a convenient way of playing microtonal and non-westen notes in a stand-alone and portable device that interfaces to existing instruments seems to be the basis for our project.

1.1 First project description

A MIDI controller that uses a grid of programmable buttons enables almost unlimited potential for exploring music in non-western tunings in a simple and visusally straightforward manner. By integrating a google nsynth in this MIDI controller, a relatively unknown domain of tone and timbre may easily be explored.

The end goal would be to have a completely programmable grid of buttons with 6 rows and 31 columns, with each row having an adjustable offset in frequency like present between the strings of a guitar. This would mean that using the same hand shapes in different positions would result in the same type of chord. There will be two buttons for cycling between the tuning sy stems, two buttons for each row for adjusting the frequency offset. There will be red-green-blue LEDs for indicating the value of the notes, with a spectrum from red to blue indicating one full octave. As a prototype, we want to be able to support the standard 12-note octave of typical western music, 17edo (equal divisions of the octave), 24edo (esentially 12-ET with quartertones within each half step), and 31edo. A suitable microprocessor/programming solution will act as the MIDI controller, accepting inputs from the button grid and making the appropriate translations for sending microtones over MIDI for output to the nsynth. The google nsynth will accept MIDI input from the software controller, where knobs will be used to adjust the amount of each instrument present, and other knobs will handle the remainder of synthesizer functions such as volume, attack, sustain, etc. We will forgo the touchscreen for design simplicity since it does not add much value to process of exploring the sound, knobs will be used instead. The most pressing task is to research a microprocessor solution to take input from a button matrix and match the microtone using a frequency shift on the midi note. Possible solutions include raspberry pi, arduino, teensy. Teensy has out-of-the-box MIDI support, while the arduino has more support.

Luckily, we already have a Raspberry Pi and CANNe Synth is written in python, so ya why would

2 Week of September 17

2.1 JT's CANNe synth paper

Time was taken to get an understanding of the canne synth, which will essentially be the heart of our project. To someone who knows little of neural nets, it helps to think of it as a compression algorithm. The autoencoder serves to create a way of representing short-time fourier transforms of audio samples in a very low dimensional space. The decoder then attempts to reconstruct an audio sample using the parameters in that very low dimensional space. CANNe Synth provides a means of generating a new sound timbre by changing around these parameters that get fed into the decoder, thus outputting a sound based on the training set but has the fingerprint of the autoencoing neural network which results in this new, unfamiliar sound.

3 Week of September 24

3.1 Software

The MIDI standard PDF was located and skimmed thru. It was discovered that microtonal notes can easily be achieved by taking advantage of the pitch modulation control function which will shift a note with a resolution of cents, which means basically any note from any non-western scale can be played. The MIDI standard works off of instrument banks, so this enables a way to implement the selection of different sound presets generated by the canne synth straight from the keyboard. When a note is pressed, a ‘note-on’ MIDI message is sent containing information about the note pressed and the strength with which it was pressed, and a ‘note-off’ message is sent upon release. Control MIDI messages are sent every time a control value changes, for example the faders on the KeyLab 88.

3.2 Tim’s presentation

3.3 Pisound

https://blokas.io/pisound/

4 Week of October 1

4.1 Software

This guide was followed in order to test using a MIDI keyboard with the Raspberry Pi: http://andrewdotni.ch/blog/2015/02/2
synth-with-raspberry-p/?fbclid=IwAR02UXpO8-V9KzqUbj2BQOyPVwTyhaboCQyyio7CL6QlEEtzU4Up-V0cDMI.
After following this, the Raspberry Pi was successfully printing out the MIDI messages received by the KeyLab 88.

5 Week of October 8

6 Week of October 15

Hardware update -issues with raspi Coding update -getting software working on mac (just talk about removing pyqt)

7 Week of October 22

7.1 Software

This week, three scripts were written to make use of the canne synth module, convert the single output file of that program to be mapped to each key on the keyboard, and to play sound when the MIDI messages are sent to the computer.

In order to test the canne synth module before making use of it, the original player.py script was modified with use of uninstallable packages such as pyqt removed. It was then discovered that the function synth.execute() of the canne synth module was the only thing that needed to be called in order to generate a new sound timbre. Thus, the first script just handled making use of this module and generating a .wav file containing the new sound timbre. This will be referred to as the New Sound File (NSF).

The next step was to map this sound file to each key of the keyboard. This was done by reading the NSF, performing a sampling rate conversion, and writing out a file using the original sampling rate. This effectively changed the frequencies and math was used in order to preserve the harmonic relationship expected from a piano. The output of this script was 88 .wav files, each representing a different note on the keyboard. However, this took literally minutes to happen and for this reason a better solution must be found.

The third script reads MIDI data in and plays the corresponding sound file. PyAudio was used for this. However, two issues were encountered. First, PyAudio would require that the duration of the note be set when the note is pressed. Because of this, a note would be triggered by a keypress but would always play with a set duration. The second issue stems from the first, which is that multiple notes could not be played simultaneously since PyAudio would pause the script when playing a note, and whichever notes are pressed while that sound is playing are ignored.

8 Week of October 29

8.1 Software

Research was done into the module pyFluidSynth in order to overcome the issues encountered with PyAudio. This proved to be an excellent solution. Unlike PyAudio, pyFluidSynth would not cause the script to pause when playing audio, would allow multiple notes to be played simultaneously, and would only play the sound while the note is being pressed.

In order to set the 'instrument,' a 'SoundFont 2' (SF2) file must be loaded into pyFluidSynth. In order to generate the SF2 file to load into the pyFluidSynth module, the NSF file was used to generate a SFZ file, which is a text markup file that is used in generating a SF2 file. The generation of a SFZ markup was done using the module 'SFZ,' and the conversion from SFZ to SF2 was done using the module 'SF2.' These modules were found online, and were written by robeto@zenvoid.org. Once the SF2 file was loading into pyFluidSynth, the appropriate sound was mapped to each key of the keyboard.

9 Week of November 5

9.1 Hardware

The poster was designed. A render was created, but this revealed that our original idea of having a grid of keyboard buttons does not afford the type of new sounds we want to afford the musician. First, the design rendered would play each note with the same pressure, which severely limits the level of expression achievable by the musician. And while microtones and non-western notes can be played with the rendered design, the notes are still quantized and would require some sort of software adjustment. Ideally, the musician should be able to play any note without restriction, like the fingerboard of a violin allows. However, this poses a conflict since using MIDI will inevitably result in note quantization. However, the MIDI frequency shift control could be used to to have any frequency which can be used to create an effective continuum of frequencies.

10 Week of November 12

11 Week of November 19

getting everything working on the raspberry pi